İSTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

COMMUNITY DETECTION IN SOCIAL NETWORKS USING PARALLEL CLIQUE-FINDING ANTS

M.Sc. Thesis by
Sercan SADİ

Department : Computer Science

Programme : Computer Science

JUNE 2010

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

COMMUNITY DETECTION IN SOCIAL NETWORKS USING PARALLEL CLIQUE-FINDING ANTS

M.Sc. Thesis by
Sercan SADİ
704071016

Date of submission : 07 May 2010
Date of defence examination: 07 June 2010

Supervisor (Chairman) : Asst. Prof. Dr. A. Şima UYAR (ITU)
Members of the Examining Committee : Asst. Prof. Dr. Şule ÖĞÜDÜCÜ (ITU)
Asst. Prof. Dr. Haluk BİNGÖL (BU)

JUNE 2010

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SOSYAL AĞLARDA TAM BAĞLI ÇİZGE ARAYAN PARALEL KARINCALAR İLE TOPLULUK BULMA

YÜKSEK LİSANS TEZİ
Sercan SADİ
704071016

Tezin Enstitüye Verildiği Tarih :	**07 Mayıs 2010**
Tezin Savunulduğu Tarih :	**07 Haziran 2010**

Tez Danışmanı :	**Yrd. Doç. Dr. A. Şima UYAR (İTÜ)**
Diğer Jüri Üyeleri :	**Yrd. Doç. Dr. Şule ÖĞÜDÜCÜ (İTÜ)**
	Yrd. Doç. Dr. Haluk BİNGÖL (BÜ)

HAZİRAN 2010

FOREWORD

I would, first, like to thank my biggest supporters in life: my family and my uncle, A. Saffet Velibeyoğlu. They were always supporting me in life and I believe that they will also be in the future. I would like to dedicate this thesis work to them as they share the biggest portion with the motivation they gave me.

Asst. Prof. Dr. Şima Uyar and Asst. Prof. Dr. Şule Öğüdücü also deserve my deep appreciation throughout my thesis period. I would like to thank them for guidance on my thesis with patience and understanding. The achievement of success could be true only by their support.

Next, I would like to express my graditutes to my collagues and my team leaders in Netaş. With the support of my friends and the understanding of my managers, I was able to continue my MSc while working. I think and totally agree that the things I learned while in academic career and Netaş combined well together; I owe a big debt of appreciation to my collagues who supported me during this period.

Thanks to God that I would not be able to come up with such achievement without his power and will.

May 2010

Sercan SADİ
Computer Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

ACO	**:** Ant Colony Optimization
ACS	**:** Ant Colony System
AS	**:** Ant System
DBI	**:** Davies-Bouldin Index
MMAS	**:** MAX-MIN Ant System
RAS	**:** Rank-based Ant System

LIST OF TABLES

LIST OF FIGURES

Page

LIST OF SYMBOLS

$\boldsymbol{G}$	: Graph
$\boldsymbol{V}$	: The vertex in the graph
$\boldsymbol{E}$	: The edge in the graph
$\boldsymbol{M_{ij}}$	: The adjacency matrix cell of node *i* and node *j* of the graph
$\boldsymbol{a_{ij}}$	: The similarity value of an edge in the corresponding community
$\boldsymbol{b_{kl}}$	: The similarity value of an edge to the outside of the corresponding community
$\Delta\boldsymbol{Q}$	: Network modularity difference
$\boldsymbol{q_0}$	: Pseudo-random proportion
$\boldsymbol{\tau_{ij}}$	: Pheromone levels between node *i* and node *j*
$\boldsymbol{\eta_{ij}}$	: Heuristic information between node *i* and node *j*
$\boldsymbol{p^{k}_{ij}}$	: The probability of ant *k* to choose next node depending on the edge in between node *i* and node *j* of the graph
$\Delta\boldsymbol{\tau^{k}_{ij}}$	: The amount of pheromone laid by ant *k* on the edge in between node *i* and node *j* of the graph
$\boldsymbol{\rho_0}$	: The pheromone evaporation rate
$\boldsymbol{\tau_{min}}$	: Minimum pheromone limit for MAX-MIN Ant System
$\boldsymbol{\tau_{max}}$	: Maximum pheromone limit for MAX-MIN Ant System
$\boldsymbol{\tau_{initial}}$	: The initial pheromone level
$\boldsymbol{r}$	: The rank of the ant
$\boldsymbol{w}$	: The weight of the best-so-far ant
$\boldsymbol{S_r}$	: The solution cost for the r-th ant
$\boldsymbol{T_k}$	: The solution found by ant *k*
$\boldsymbol{T_{bs}}$	: The solution found by the best-so-far ant
$\boldsymbol{\alpha}$	: The effect of pheromone level
$\boldsymbol{\beta}$	: The effect of pheromone level
$\boldsymbol{C_i}$	: *i*-th clique in the graph
$\boldsymbol{x_{ij}}$	: The relevance value of edge in between node *i* in the clique-node to node *j*
$\boldsymbol{y_{ij}}$	: The relevance value of edge in between node *i* and node *j* of the clique-node
$\boldsymbol{e_{kl}}$	: The relevance of the newly formed edge in between node *k* and node *l*
$\boldsymbol{\xi}$	: The local pheromone update parameter for Ant Colony System
$\boldsymbol{m}$	: The number of ants
$\boldsymbol{N^{k}_{i}}$	: The neighbourhood of nodes in ant *k*'s journey

COMMUNITY DETECTION IN SOCIAL NETWORKS USING PARALLEL CLIQUE-FINDING ANTS

SUMMARY

Constantly increasing popularity of Internet attracted people to share and collaborate more information with the rest of the world. This phenomenon motivated many disciplines to expand their research areas onto social networks which are also constantly growing parallel to the advent of Internet. The growth of the social networks with help of Internet also led research areas to search of community structures to be established on those networks. Community structures can be established depending on the interactions between the network elements and the detection of those structures became popular in the last years.

The basis of community detection, the community structure, can be defined with the density of interaction in between the network members of the corresponding network. In graph theory, networks are represented with graphs, where the network members are nodes/vertices and the interactions in between them are the edges of the graph. Thus, the definition of community can be formed as follows: a group of nodes which possess higher density of edges in between and lower density of edges going other nodes out of that group can be named as community.

There are many community detection methods emerged with the popularity of the subject. The popular ones can be named as hierarchical clustering, spectral bisection and fast greedy community detection method based on modularity maximization. The *modularity* is a quality assessment parameter proposed for community detection and its widely used on aforesaid community detection tools as an indicator of clustering quailty.

Inspite the fact that there is a lot of improvement on the community detection methods (i.e. on time complexity), they still suffer from high computational costs and ineligible scalability on large-scale network graphs. In this thesis study, we propose a novel method to reduce the graph to a maintainable size while preserving its quality based on modularity. With the algorithm we propose, the community detection tools will be less affected from the scalability and computational cost problem on large-scale social networks.

As the basis of our reducing algorithm, we used the clique scheme which can be shown as the basic structure of a community on network graphs, along with clans and plexes. The *clique* is the fully connected subgraph where the almost fully connected subgraph is named as *quasi-clique* in graph theory. In the thesis study, we accept the quasi-cliques as the basis of communities and try to find all possible quasi-cliques in the network graph with an nature inspired optimization tool: Ant Colony Optimization (ACO). The Ant Colony Optimization technique uses ants to search the optimum solution in a given problem. They use pheromones to favor the overall optimum solution each iteration and lay the pheromones on the solution path for ants to follow the path on the next iterations. Ants search for all possible quasi-cliques in

their journey on the graph to construct the best solution which leads to better reducement with minimum quality loss.

The steps of our proposed algorithm are defined as follows:

1. Depending on the size of the network graph, especially on large-scale graphs due to concerns on computational cost, a snowball sampling method is applied to whole graph to create subgraphs on the original graph. Each subgraph will be handled by threads in parallel for further processing.
2. ACO models are run on each snowball thread in parallel. The used ACO models in the process are Ant Colony System (ACS), Max-Min Ant System (MMAS) and Rank-based Ant System (RAS). The ants on find the best collection of quasi-cliques one each subgraph. The cliques are intended to be fully connected, however, regarding the relaxation threshold defined for our thesis study, the connectedness of the clique can be relaxed upto a threshold, which will in return allow to collect quasi-cliques on the journey. The best ant is than chosen with the highest total score gained, depending on its clique collection's quality.
3. As the clique collection found by the ants intersect a node in between each clique of the collection, it should be fixed. This problem is called overlapping and its fixed right after the ACO step results with a clique collection. The shared nodes are assigned to clique with higher number of nodes that shares it.
4. The fixed cliques are transformed into a single node, called *clique-node*, which will be used with other clique-nodes and unassigned nodes in graph transformation phase. In this phase, the exisiting edges are removed and new edges are created to connect new nodes of the reduced graph, with assigned weight values based on a weighting scheme derived from the concept of *edge-betweenness.*
5. On the last phase, newly emerged reduced graph is processed with a fast greedy community detection method and the results are compared with the original graph's results.

We run our experiments on several medium-scale and large-scale social network graphs as well as some benchmarking datasets. The experiments produced results on *number of cliques found, total score achieved, number of nodes and edges in the reduced graph, number of communities found, overall modularity of the graph* and *Davies-Bouldin Index value of the* graph. The results of the experiments show that the ACO models do not differ significantly on clique quality and the overall solution quality. Modularity values seems to be preserved on the reduced graph compared to the original graph, while Davies-Bouldin Index, which is used as a cluster validty tool, also validated the results of clustering on reduced graph and the original graph. In addition, we monitored a reducement of 50% on nodes and edges on the original graph, which will led to a improvement of time complexity *O(E.VlogV)* of the used fast greedy algorithm to *O((E.V/4)log(V/2))*, when used with our preprocessing.

Consequently, we recommend the use of each ACO models in the process to optimize the computational costs and scalability of the any community detection method used with the preprocessing algorithm proposed in this thesis.

SOSYAL AĞLARDA TAM BAĞLI ALT ÇİZGE ARAYAN PARALEL KARINCALAR İLE TOPLULUK BULMA

ÖZET

İnternet ağının sürekli artan popülaritesiyle birlikte, insanlar daha çok bilgiyi ağ üzerinden dünyanın geri kalanıyla paylaşmaya ve geliştirmeye başladılar. Bu fenomen birçok farklı disiplini, İnternetin gelişimine eşzamanlı ve sürekli bir şekilde büyüyen sosyal ağlar üzerinde araştırma faaliyetlerini genişletmeye yüreklendirdi. İnternetin yardımıyla sosyal ağların büyümesi araştırma alanlarını bu ağ çizgeleri üzerinde topluluk yapısı aramaya yöneltti. Ağ elemanları arasında oluşan/varolan etkileşimlere dayalı olan topluluk yapıları ve bunların tespiti, son yıllarda popüler olmaya başladı.

Topluluk arama yöntemlerinin temeltaşı, topluluk yapısı, verilen ağ çizgesinde bulunan ağ elemanları arasındaki etkileşimin yoğunluğu ile tanımlanabilir. Çizge teorisinde ağ yapıları çizge ile temsil edilir; ağ elemanları düğüme, elemanlar arasındaki etkileşim/yakınlık göstergesi ise ilgili iki düğüm arasındaki ayrıta karşılık düşer. Bununla ilintili olarak çizgelerdeki topluluk yapıları şu şekilde tanımlanabilir: birbirileri arasındaki ayrıt sayısı gruba dahil olmayan diğer ayrıtların sayısına göre fazla olan düğüm grupları *topluluk* olarak adlandırılır.

Konunun popülerliğinin artmasıyla birlikte birçok topluluk bulma algoritması ortaya çıkmıştır. Bunlardan popüler olanları, aşamalı kümeleme, spektral bölümleme ve birimsellik enbüyütme prensibi ile çalışan hızlı aç gözlü topluluk bulma algoritmasıdır. *Birimsellik*, topluluk bulma yöntemi için önerilen bir kalite analizi aracıdır ve birçok topluluk bulma algoritması tarafından kümeleme kalitesini ölçmek için kullanılır.

Topluluk bulma algoritmalarındaki birçok yapılan iyileştirmeye (örn. zaman karmaşıklığı) rağmen, bu algoirtmalar işlem maaliyetleri ve büyük ölçekli ağ çizgeleri üzerinde ölçeklendirilme sorunu yüzünden olumsuz yönde etkilenmektedir. Bu tez çalışmasında, eldeki çizgeyi ölçeklenebilir bir boyuta indirgeyebilen ve bu indigeme sonucunda birimselliğe dayanan kalite analizinden minimum kayıpla çıkan bir yöntem önerilmektedir. Önerilen bu yöntem ile topluluk algoritmaları, işlem maaliyetleri ve ölçeklendirilşme sorunundan daha az etkilenecektir.

Önerdiğimiz algoritmanın temeli olarak, klanlar ve pleksler gibi, ağ çizgelerindeki toplulukların yapıtaşı olarak kabul edilen tam bağlı alt çizgeler kullanılmıştır. *Tam bağlı alt çizge (hizip)*, bütün düğümleri arasında en az bir ayrıt olan düğüm kümelerine denir. Aralarında ayrıt olmayan düğüm çifti sayısının genele oranla çok çok az olduğu alt çizgelere ise *yarı-bağlı alt çizgeler* adı verilir. Bu tez çalışmasında yarı-bağlı alt çizgeler topluluk yapılarının yapıtaşı olarak ele alındı ve çizgelerdeki bütün olası yar-bağlı alt çizgelerin, doğal esinli bir eniyileştirme aracı olarak Karınca Kolonisi Eniyileştirmesi yöntemi, bulunması amaçlandı. Karınca Kolonisi Eniyileştirmesi tekniği, karıncaları kullanarak problem üzerindeki en iyi sonucu bulmaya çalışır. Karıncalar en iyi yöntemi belirlemede yön gösterici olması için her

çzöüm üretme adımında feromon salgılarlar ve bu feromonu problemde ürettikleri çözüm yolu üstüne, bir sonraki adımda başka karıncalar tarafından izlenebilmesi için bırakırlar Karıncalar çizge üzerindeki olası bütün yarı-bağlı alt çizgeleri bulup en iyi çözümü üreterek, çizge üzerinde minimum kalite kaybı ile indirgeme yapmaya çalışırlar.

Önerilen algoritmanın adımları aşağıdaki sıralanmıştır:

1. Çalışılan ağ çizgesinin boyutuna bağlı olarak, özellikle büyük ölçekli çizgelerde doğabilecek işlem maaliyeti sorununa önlem amaçlı, çizgeyi daha küçük alt çizgelere bölümlendirmek için kartopu örneklemesi yapılır. Oluşan her alt çizge üzerinde, bir sonraki adımlar için biriryle paralel işleçler çalışır.

2. Oluşturulan her kartopu işlecinde Karınca Kolonisi Eniyileştirmesi modelleri çalışır. Bu adımda kullanılan yöntemler sırasıyla Karınca Kolonisi Sistemi, Ençok-Enaz Karınca Sistemi ve Rütbe-bazlı Karınca Sistemi'dir. Karıncalar her alt çizgede en iyi yarı-bağlı alt çizge listesini oluşturmaya çalışırlar. Çizgelerin tam bağlı olması amaçlanır, fakat bu tez çalışmasında önerilen bir eşik değeri ile çizgede belli oranda ayrıtın eksik olmasına izin verilir ve işlem boyunca yarı-bağlı alt çizgeler de listeye eklenir. Çözüm listesindeki tam veya yarı-bağlı alt çizgelerin kalitesine bağlı olarak en iyi puana sahip karınca, en iyi karınca seçilir.

3. En iyi karıncalar tarafından oluşturulan alt çizge listesindeki çizgeler, işlemin doğası gereği bir düğümü paylaşırlar. Bu sorun üstüste binme olarak adlandırılabilir ve bu aşamada düzeltilir. Düzeltme işleminde paylaşılan düğüm en çok düğüme sahip olan ve bu düğümü paylaşan alt çizgeye verilir.

4. Düzeltilen alt çizgeler tek bir düğüm haline dönüştürülür; diğer düğüm grupları ve atanmamış düğümlerle birlikte çizge dönüştürme işleminde kullanılacak bu yeni düğüme *çizge-düğüm* denir. Bu adımda, işlenmemiş ana çizgedeki tüm ayrıtlar silinir ve ayrıt-arasındalık kavramından esinlenerek belirlenen ağırlık değeri ile çizge-düğüm ve atanamamış düğümler arasında yeni ayrıtlar yaratılır.

5. Son adımda yeni oluşturulan indirgenmiş çizge, bir hızlı aç gözlü topluluk bulma algoritması ile işlenir; sonuçlar işlenmemiş çizgenin sonuçları ile karşılaştırılır.

Popüler kıyaslama verikümeleri ile orta ve büyük ölçekli sosyal ağ verikümeleri üzerinde testlerimizi koştuk. Testlerimizin ürettiği, karşılaştırma için kullanılan değerler, *toplam tam veya yarı-bağlı alt çizge sayısı, toplam puan, indirgenen çizgedeki düğüm ve ayrıt sayısı, toplam topluluk sayısı, genel birimsellik* ve *Davies-Bouldin İndeksi*'dir. Sonuçlar, tüm Karınca Kolonisi Eniyileştirme modellerinin, bulunan tam veya yarı-bağlı alt çizge ve genel çözüm kalitesinde biribirine yakın olduğuna işaret etmektedir. Çizge indirgenmesi sonrasında hesaplanan birimesllik değerlerine göre kalitenin korunduğu gözlenmiş ve Davies-Bouldin İndeksi'ne göre de kümeleme kalitesi açısından da en alt seviyede kayıp olduğu doğrulamıştır. Bunlara ek olarak, çizge üzerinde düğüm ve ayrıt sayısı bkaımında %50'ye varan bir azaltılma gözlemlenmiş, bu sonucun da, önerilen önişleme yöntemiyle beraber kullanıldığında, *O(E.VlogV)* zaman karmaşıklığına sahip topluluk bulma algoritmasının karmaşıklığını *O((E.V/4)log(V/2))* değerine indirdiği hesaplanmıştır.

Sonuç olarak, bu tez çalışmasında sunulan önişleme yönteminin, kullanılacak herhangi bir topluluk bulma yönteminin, işlem maaliyetleri ve ölçeklendirilme

sorunun en aza indirgenmesi adına, bahsi geçen 3 Karınca Kolonisi Eniyileme yöntemlerinden biri seçilerek, ilgili topluluk bulma yöntemi ile beraber kullanılmasını tavsiye ederiz.

1. INTRODUCTION

The continuous growth of Internet, which makes information using/sharing and collaborating easier for people, also allows the attractiveness of social networks as a research topic in many different disciplines grow in parallel. Depending on the frequency/density of interactions/similarities between each network members, community structures might be established in those social networks. Detection of these community structures is a popular research topic.

The definition of community in a social network, from computer science perspective, can be given as follows. Nodes/vertices represent the members of a given network while edges in between the nodes represent the relevance/interaction/similarity between the corresponding nodes. With regards to latter definition of network graph, *communities* are defined as the group of nodes with higher density of edges in between, when compared to the outward edges (edges from the community nodes to the outer nodes which reside out of that community) [1]. A term, proposed in the context of community detection on network graphs by Girvan and Newman [2], the *modularity* is the density indicator of whole communities in a given network, used as a quality metric.

There are many variants proposed for community detection, which use different approaches such as greedy approaches [3], or hierarchical clustering on the given social network [4]. However, large-scale social networks cause scalability problems related to increasing computational costs when these community detection methods are used. The computational complexity of these community detection methods comes from these two parameters: number of nodes and number of edges in the given network. In this thesis work, we propose a novel method which enables those community detection methods to process effectively on large-scale social networks. The proposed method reduces the size of the network, which reduces the execution times of the community detection methods on large-scale networks which improves methods' scalability while preserving the solution quality.

In our approach, the base element that forms communities, *cliques*, are used to detect community structures. *Clique*, a graph theory concept, can be defined as a fully connected subgraph, whereas an almost fully connected clique is named as a *quasi-clique*. In [5], quasi-cliques are accepted as the basis of communities. *Ant Colony Optimization* (ACO) techniques are used in literature [6] to search for cliques in a given graph. We used a modified version of an ACO based maximum clique search algorithm [6] to find the quasi-cliques of all possible sizes in the given graph. *Overlapping cliques* (cliques which share node with other cliques) are corrected after ACO step. The resulting cliques, named *metanodes* or *clique-nodes* in this study, are used in graph transformation step. Graph transformation step is required to shrink the original network graph to a manageable size for community detection methods. In this step, connections between the individual nodes belonging to each clique are used to form new edges between clique-nodes. At the last step, a traditional community detection method [7] is used on the transformed graph for community detection. The aforementioned approach is implemented and the experiments are run on benchmark social networks commonly used to compare results of community detection approaches [8]. We use the snowball sampling method [9], which is a technique to create samples starting from a random instance and growing like a snowball by adding the neighbors of that instance to the pile, to generate these subgraphs and we run the ACO-based clique finding technique on each one in parallel, which allowed us to run our experiments for larger-scale social networks, which is also used in [10].

This thesis is structured as follows: first, in Section 2, we give a problem definition for community detection on given networks and we present related work in social networks and community detection. Following with Section 3, we explain the ACO technique and give details about the two ACO variants we use in this study, namely the Ant Colony System (ACS), the Max-Min Ant System (MMAS) and the Rank-based Ant System (RAS). Then, we present our proposed approach to find quasi-cliques in a graph for community detection in Section 4. Section 5 shows our experimental results and our analysis of these results. Finally, Section 6 concludes the paper and provides directions for possible future work.

2. COMMUNITY DETECTION PROBLEM

2.1 Problem Definition

The community detection problem on social networks, while considering many definitions proposed in the literature, can be defined and formulated as follows: Nodes or vertices are represented with set *V* while edges in between those vertices, which show the pair wise connections between the individuals (similarities/relevancies between two individuals), are represented with set *E*. Graph *G=<V, E>* is a model of a social network. With the given graph definition, a *community* can be defined as a subgraph in a network graph that has a higher density of edges in between its members and a lower density of edges from its members to those outside the subgraph.

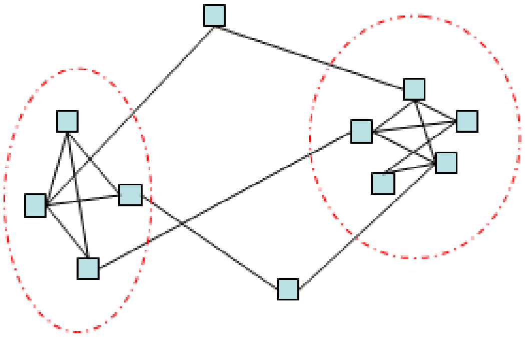


Figure 2.1 : Community definition

The social network graph is represented by an adjacency matrix *M* of *NxN* where *N* is the number of nodes in the corresponding graph. An adjacency matrix cell M_{ij} is the indicator to an edge between the nodes *i* and *j* of the graph. The value of the cell will be 0 if there is no connection (no similarities or interaction) between the corresponding nodes; the value will be 1 or a positive real value depending on the unweighted or weighted edges on the corresponding network graph. The problem of community detection is to find k number of communities in a given network graph, such that each community satisfies Eq. (2.1):

$$\begin{aligned} &\sum_{i\in K}\sum_{j\in K} a_{ij} > \sum_{k\in K}\sum_{l\notin K} b_{kl} \\ &b_{kl} = b_{lk}, \\ &i,j,k,l \in \{1,2,\ldots,N\} \end{aligned} \qquad \textbf{(2.1)}$$

where a_{ij} is the similarity (a relation/relevance indicator value in real number form) value of an edge in the community K and b_{kl} is the similarity value of an edge to the outside of that community K.

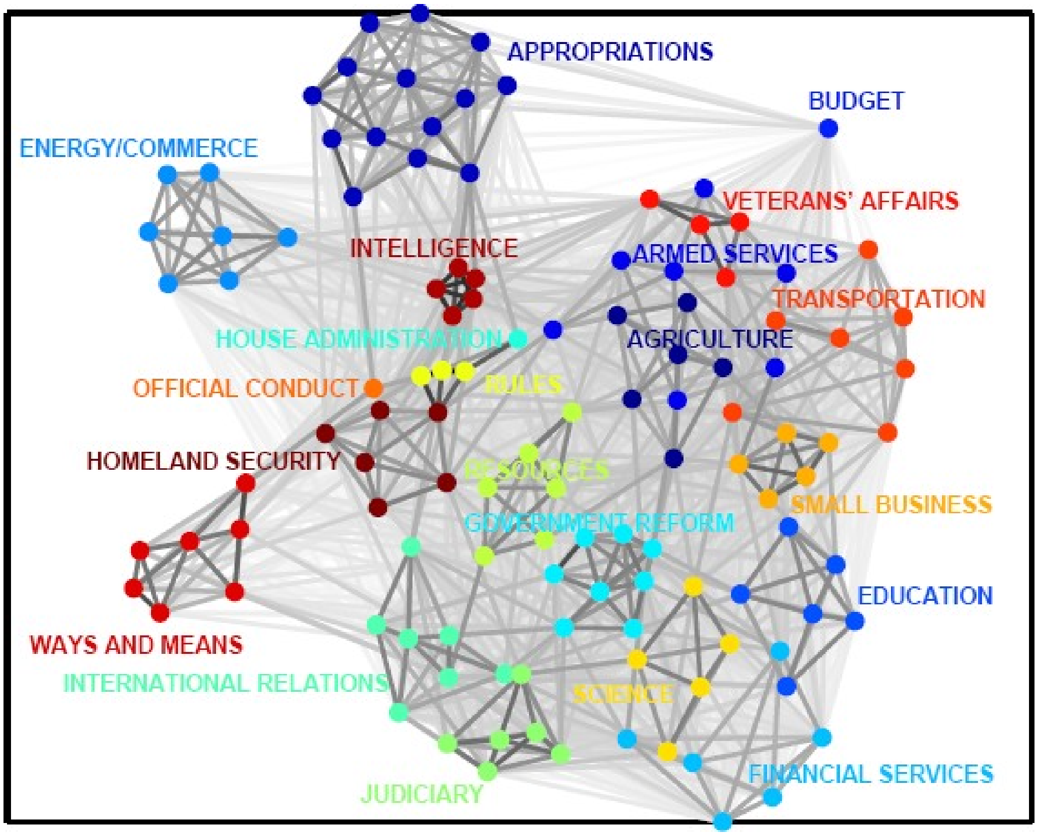


Figure 2.2 : An example community structure

Essentially, the community detection problem is a type of clustering problem. However, types of the network data used in the problem lead to significant differences. In the original clustering problem, a similarity or distance matrix for the given network data is enough to apply clustering methods (i.e. *k*-means, hierarchical or spectral clustering). On the other hand, discrete network data (i.e. biological or social networks) are different from the above menitoned network data; they are large-scaled compared to the other types (real-world data and commonly have a power law degree distribution), and contains data patterns to be detected by graph algorithms (i.e. cliques). As a result, the community detection problem is based on different parameters when compared to original clustering, which can be named as edge-betweenness, network modularity and so on.

2.2 Related Literature

Cohesive groups like cliques, clans or plexes, can be the definition of communities. There are many approaches in literature which use those cohesive groups as the basis of their detection method. Among those approaches, Donetti and Munoz proposed a hierarchical clustering approach, based on detection of larger communities using Laplacian eigenvectors as a similarity measure on a given network graph [4]. The essence of the approach lies on the division operation used to establish communities, while the vectors are re-calculated as long as the division operation continues. Despite the fact that the initial number of communities on the given graph is not required by the method, the termination condition for the division process cannot be optimized to come up with the best clustering result on the corresponding graph.

The *Network Modularity* term, proposed by Girvan and Newman [11], is introduced by a divisive approach structured on elimination of edges from the network graph based on the betweenness values. The betweenness used here is based on *edge betweenness* where weights are assigned to edges, which are stationed on the shortest path between pairs of nodes. The edge betweenness value increases in parallel with the number of shortest paths on that edge. “Q”, the network modularity, is the ratio of in-community edges to the randomly chosen edges on a network subgraph. Network Modularity, Q, takes on values in between 0 and 1. The value depends on the clustering measure on the given graph. A close to 1 value means the communities in the graph have fewer connections to the outside of that cluster when compared to its inward connections; likewise, a close to 0 value means the opposite. An optimized Q value helps to find a better division on a given network graph, although the performance loss on large-scale network graphs is still a drawback for the proposed algorithm. Considering the performance of corresponding method, Radicchi proposed a similar edge clustering algorithm with better performance [12].

A more enhanced, fast greedy clustering method, based on modularity maximization, is proposed by Clauset, Newman and Moore [3]. Clustering continues by merging nodes with maximum ΔQ and stops when ΔQ results are negative. Although Wakita and Tsurumi [7] came up with an optimized version of this method, there are still concerns on performance and solution quality for large-scale graphs.

A different clustering algorithm, proposed by Palla et al. [13], uses cliques as a basis of the detection similar to the approach in this thesis. In the approach, called *k-clique percolation*, an edge probability equation is proposed to find a suitable *k* value for the *k-cliques* to be created. Once a suitable value is found, a giant component is searched by attaching k-cliques one-by-one. The algorithm is said to be successful with better performance on overlapping community detection on Erdos-Rényi (ER) random graphs.

As the popularity of community detection increases, different approaches are being proposed. Nature inspired approaches are also used in this manner. A genetic algorithm proposed by Pizzuti can be given as an example [14]. The algorithm uses a fitness function for the diversification of node groups and establishes communities. ACO techniques are also used for community detection. The study of Liu et al. [15] proposes an ant clustering technique based on Enron's mail network communities. In the preliminary study of the thesis, described in [8] and [10], we also used an ACO technique. However, unlike in [14], ACO is not used for clustering. We used ACO to determine cliques, which will then be used as vertices in a reduced graph. A regular clustering based community detection algorithm is then applied on this reduced graph. By doing this, we aimed to overcome the performance loss of community detection methods on large-scale networks. For optimization of the study and structuring of the thesis, we further modified our approach to work in parallel on subgraphs of the original network graph as in [10], which were created using snowball sampling. We also modified the algorithm to search for quasi-cliques, which will be described in detail in Section 4.

There are also similar graph reducing studies called as "graph coarsening", which is a part of "Multi-level Graph Partitioning". Their main process is based on 3 sub-processes: coarsening, partitioning and un-coarsening. A detailed comparison on the schemes which Multi-level Graph Partitioning use is given in [16], as well as an evolutionary approach proposed in [17]. Even though the coarsening part is similar, the approach is not fully applicable to our algorithm as it mostly works on graphs with weights on both edges and nodes. Following to that, the partitioning is based on balanced partitions on the graphs, unlike community detection which is based on clustering in search for a common trait.

3. ANT COLONY OPTIMIZATION

ACO, one of the most commonly used swarm intelligence techniques in literature, is based on the behavior of real ants. ACO was first introduced by Marco Dorigo in his PhD thesis [18]. In the real world, ants (initially) wander randomly, and upon finding food return to their colony, while laying down a special chemical called the *pheromone.* This is used to communicate with other ants. If other ants come across a path with pheromones on it, they are likely to follow the trail, returning and reinforcing it if they also find food along the same path.

The basic ACO algorithm is given below. An ACO iteration consists of the solution construction and pheromone update stages. In each iteration, each ant in the colony constructs a complete solution. Ants start from random nodes and move on the construction graph by visiting neighboring nodes at each step.

Algorithm 1 Basic ACO Outline

```
set ACO parameters
initialize pheromone levels
while stopping criteria not met do
    for each ant $k$ do
        select random initial node
        repeat
            select next node based on decision policy
        until complete solution achieved
    end for
    update pheromone levels
end while
```

An ant k chooses the best neighbor with a probability of q_0. Otherwise, the next visited node is determined using a stochastic local decision policy based on the current pheromone levels τ_{ij} and heuristic information η_{ij} between the current node and its neighbors with a probability p^k_{ij} as calculated in Eq. (3.1); where α and β are integer values to define powers of pheromone levels and heuristic information, N^k_i is the neighborhood of nodes for ant k's journey.

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, j \in N_i^k \qquad \textbf{(3.1)}$$

Pheromone trails are modified when all ants have constructed a solution. First, the pheromone values are evaporated by a constant factor on all edges. Then, pheromone values are increased on the edges the ants have visited during their solution construction. Pheromone evaporation and pheromone update by the ants are implemented as given in Eq. (3.2) and Eq. (3.3) respectively,

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} \qquad \textbf{(3.2)}$$

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \qquad \textbf{(3.3)}$$

where $0 < \rho \leq 1$ and $\Delta\tau^{k}_{ij}$ is the amount of pheromone deposited by ant *k*.

ACO has been applied successfully to many combinatorial optimization problems, such as routing problems, assignment problems, scheduling and sequencing problems and subset problems, etc. Ant System (AS) is the first implementation of ACO algorithms, and has been the basis for many ACO variants. There are many successful AS variants in literature. Among the most commonly used variants, the elitist AS, rank-based AS, the MAX-MIN AS (MMAS), the ant colony system (ACS), the best-worst AS, the approximate nondeterministic tree search, and the hyper-cube framework can be mentioned [19]. MMAS and ACS are shown to be good both in solution quality and also in solution speed for the example cases in [19]. Therefore, we also use them in this study. In addition to above variants, RAS is also used in our study to observe the differences with MMAS. MMAS, RAS and ACS are among the approaches which can be considered as direct variants of AS, since they both use the basic AS framework. The main differences between AS and these variants are in the pheromone update and pheromone management details. The AS algorithm implements the basic ACO procedure detailed above. The following paragraphs explain the differences between the selected ACO variants and AS. For further details see [19].

3.1 The Ant Colony System (ACS)

The ACS [19] differs from AS in three main points:

- First, a pseudo-random proportional action choice rule is used, which allows the exploitation of the ants' search experience.
- Secondly, pheromone evaporation and deposit is applied to the edges of the best-so-far solutions.
- Finally, a local pheromone update, which includes evaporation, is applied each time an ant passes through the corresponding edge. This favors exploration over exploitation.

At the end of each iteration in ACS, the pheromone trails are again updated similar to in AS, but the pheromone trail updates, both evaporation and new pheromone deposit, are implemented only for the edges belonging to the best-so-far solution.

3.2 The MAX-MIN Ant System (MMAS)

The *MAX-MIN Ant System (MMAS)* [19] has four major differences from AS:

- First, the pheromone update is allowed for the iteration-best, that is the ant with the best solution for that iteration, or best-so-far ant, that is the ant with the best solution for all iterations, throughout the runs.
- Secondly, pheromone limits in an interval [τ_{min}, τ_{max}] is defined to prevent stagnation on local optimum.
- Thirdly, edges are initialized with upper pheromone limits to favor exploration over exploitation in the beginning of the run.
- Finally, the pheromone trails are reinitialized when the solution does not improve for a number of iterations or stagnation occurs.

Pheromones are deposited on the edges according to the equations as given for AS above. The difference is that the ant which is allowed to add pheromone may be either the best-so-far or the iteration-best ant. Commonly in MMAS implementations, both the iteration-best and the best-so-far update rules are used alternatively.

3.3 The Rank-Based Ant System (RAS)

Rank-based Ant System (RAS) [19] is an improved version of the original Ant System (AS). The amount of the pheromone, which selected ants deposit on the trail, decreases over time with respect to their ranks of their solution. The ranks are decided after the solutions are ordered by their solution quality. Each ant deposits its pheromone according to their weight related with rank *r*. In each iteration, the best-so-far ant and the remaining (*w*-1) best ant deposit their pheromones. The *r*-th best ant will have a weight *max* (0, *w*-*r*), while the best-so-far ant's weight is *w*. Eq. (3.4) is the pheromone deposit rule for RAS, where S_r denotes the solution cost for *r*-th best ant.

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{r=1}^{w-1}(w-r)\cdot\Delta\tau_{ij}^{r} + w\cdot\Delta\tau_{ij}^{bs},$$
$$\Delta\tau_{ij}^{r} = \begin{cases} edge(i,j) \in Tr \Rightarrow 1/Sr \\ edge(i,j) \notin Tr \Rightarrow 0 \end{cases} \tag{3.4}$$

3.4 ACO for the Maximum Clique Problem

For the maximum clique version of ACO, each ant is placed on a random node of the given graph *G*=<*V, E*> where *V* is the set of nodes and *E* is the set of edges between them. Ants lay pheromones on the edges of the cliques they find through their walk. Ants are forced to visit a node only once in their journey, by keeping a tabu list for each ant. This list contains all the nodes in the ant's trajectory until it gets stuck or it finds a feasible solution. In such a case, the ant restarts its journey per request and its tabu list is reset. Each ant chooses its next node based on the probabilistic state transition rule, given in the previous subsection that uses pheromone values and heuristic information as components. Also note that nodes are chosen by the ant if they establish a clique with nodes the ant visited: next node to be selected should have connections with all nodes in the clique the ant has created. After each ant applies the same rules and creates a solution, pheromone update is performed, based on the used ACO variant. The pheromones are deposited on the edges of the found cliques. For further details on the ACO for Maximum Clique problem, please refer to [6].

4. COMMUNITY DETECTION USING ACO

In [8], we proposed an ACO-based technique for community detection. In this thesis, we improve our approach through the below steps.

1. Given network graph is divided into subgraphs through snowball sampling method.
2. ACO techniques are applied in parallel to each snowball sample in the search of quasi-cliques.
3. Overlapping cliques are fixed.
4. Fixed subgraphs with non-overlapping cliques are combined and transformed into a graph smaller than the original graph. The graph re-creation is based on the concept of *betweenness*, to construct new edges for the graph.
5. The resulting, transformed graph is processed with a community detection algorithm to find possible community structures in.

In the preliminary study, we used the above steps to reduce the network graph, using ACO to search for fully connected cliques. As an enhancement on the protoype version of the algorithm, we modified the search dynamics and the reconstuction phase as well as parallelization of the method. We are able to reduce the size of the network even more, through relaxing the fully connected clique search constraint and modification to search for quasi-cliques. Through sampling and then parallelization, we are able to search on the original network in parallel. This increases the scalability of our proposed approach.

4.1 Snowball Sampling for Creating Subgraphs

In the preliminary study, ants traversed the whole graph to search for cliques. As an improvement to shorten the execution times on large-scale network graphs, a sampling method is performed on the whole graph to create subgraphs which then allowed us to use ACO techniques on each of the subgraphs in parallel.

First, the number of parallel ACO threads to be run on the network graph is determined with respect to the size of the corresponding graph. The decided value is given as a parameter to our method. Snowball sampling is performed on the given graph to create subgraphs for each of the threads. Each snowball agent is placed on a random node in the graph and it walks on the graph by adding neighbor nodes, until the snowball can't grow any more. Then each snowball becomes a subgraph on which an ACO thread executes. In the process, if the snowball has fewer nodes then the threshold limit, which is chosen as the number of ants, then the snowball is discarded and started again while there are available unvisited nodes in the graph. The snowball sampling algorithm is shown below.

Algorithm 2 Snowball Sampling

```
initialization of snowball sample memory
while there is an unfinished snowball do
    for snowball thread $t$ do
        if snowball is not successful then
            select random initial node
        if there is a node available then
            select next connected and unvisited node
            add selected node and the edges
        else
            if number of nodes in snowball is above threshold then
                mark snowball successful
            else
                mark snowball not successful
                release acquired nodes and edges
    end for
end while
```

4.2 Ant Colony Optimization for Finding Quasi-Cliques

4.2.1 Clique Finding Approach

The Maximum Clique Finding Problem is the basis of the ACO aided search for quasi-cliques in our solution. The original ACO based approach for the mentioned problem is proposed by Fenet and Solnon [6], which uses an ACO variant, MMAS. In the original approach, ants try to find the possible maximum clique in the given network graph. In our approach, ants try to find all quasi-cliques on their path as well as the possible maximum clique. An ant moves along its way while constructing its cliques and starts a new clique when there is no eligible neighbor node left to add to the current clique. In this approach, ACS and RAS are also used as ACO variants besides MMAS.

Different from the preliminary work on clique search, the search mechanism is relaxed with a threshold value for connectivity. Quasi-cliques, which are actually almost fully connected subgraphs, are dependent on that threshold value defined for connectivity. An ant will try to find cliques on its way but it may also allow quasi-cliques which satisfy the beforementioned threshold value. During the journey of the ant, the next candidate node can be added to the current clique to form a quasi-clique if the ratio is below the threshold value, which can be defined as the ratio of unconnected nodes against the next node to number of nodes in the current clique.

4.2.2 Pheromone Trails and Heuristic Information

Ants decide to move on the next eligible node in their journey. The selection of the next node depends on 3 parameters. The node must be unvisited, which is controlled by a tabu list deployed to each ant. The second and the third parameters are the pheromone level and the heuristic information, which will be described in this section.

In the solution, pheromones are deposited on the edges of the given network graph. Thus, the pheromone levels are represented with a two dimensional array, whose cells are mapped to the edges of the graph. The pheromone level on an edge can be symbolised as τ_{ij}. Higher pheromone levels on the edges attract ants to move through them to add the nodes terminating on the corresponding edges. Higher pheromone leves indicate the possibility of finding better cliques are on the trail. The amount of pheromone laid on the edges are proportional to the quality of the solution. Pheromones are initialized on the edges at the beginning of the process. The pheromone levels are set to the same value for each ant, but the differentiation is achieved by the addition of the heuristic information to the selection process. The heuristic information, η_{ij}, is the average of the degrees of the candidate nodes for ants' choice on the journey. Ants use both heuristic information and pheromone combined together, to select the next best node to construct a series of quasi-cliques. The combined value for the pheromone is called the total information and is defined as $\tau^{\alpha}{}_{ij}$. $\eta^{\beta}{}_{ij}$, where α and β are the constants to adjust the weights of pheromone level and heuristic information.

A *popular neighborhood* tour is executed beforehand for pheromone initialization. The tour creates a popular neighborhood list. Each row in the list corresponds to a selected node and the columns correspond to the nodes connected to that node, sorted in decreasing order of degrees of the nodes. This popular neighborhood list is used in the heuristic approach for each node in the sub-graph. The pheromone limits differ for each ACO variant. The following equations show the pheromone initialization for each model. The equations (4.1), (4.2) and (4.3) are for ACS, MMAS and RAS respectively.

$$\tau_{initial} = n.\text{pn\_tour}() \qquad \textbf{(4.1)}$$

$$\begin{aligned} \tau_{\max} &= \rho.\text{pn\_tour}(), \\ \tau_{\min} &= \tau_{\max}.(2n)^{-1} \end{aligned} \qquad \textbf{(4.2)}$$

$$\tau_{initial} = \rho.\text{pn\_tour}() \qquad \textbf{(4.3)}$$

Pheromone limits are directly related to the number of ants used in the solution for the τ_{min} value and the best-so-far ant's score achieved at the "popular neighborhood" tour for the τ_{max} value (τ_{initial} value for ACS and RAS). The pheromone levels on all edges are set to the τ_{max} value to favor exploration in the beginning of the run. The function *pn_tour()* returns the total score gained by the scout ant's clique collection constructed on the "popular neighborhood" tour, *n* is the number of nodes in the graph and *ρ* is the evaporation rate. The pheromone constants *n* and *$(2n)^{-1}$* are chosen as in TSP problems.

The pheromone is globally distributed on the edges of the cliques found by the ants and the amount is dependent on the solution quality defined as the *score* of ant. A scoring system evaluates the score achieved by an ant, based on the cliques it has found so far, as shown in (4.4)

$$\begin{aligned} midsum &= \sum_{l=1}^{nbCliques} (vertices^{2}(C_l) + edges(C_l)), \\ \text{score}(ant_k) &= \frac{midsum}{nbCliques} \end{aligned} \qquad \textbf{(4.4)}$$

where C_l is the current clique found by the ant and *nbCliques* is the number of cliques found by the current ant. *vertices()* and *edges()* functions give the number of vertices and edges in the current clique accordingly. Following that, the pheromone difference can be calculated as in (4.5).

$$\Delta \tau(ant_k) = 1 - (\text{score}(ant_k))^{-1} \tag{4.5}$$

The pheromone update procedures differ according to the ACO version used in the process. There are update procedures mentioned here: global update, weigthed global update and local update.

Algorithm 3 Global Pheromone Update

```
for each ant $k$ do
    for each clique $C_l$ do
        for each edge $ij$ do
            $\tau_{ij} \leftarrow \tau_{ij} + \left[ \Delta\tau(ant_k) \cdot \frac{vertices(C_l)}{m\_vertices(ant_k)} \right]$
        end for
    end for
end for
```

In Algorithm 3, global pheromone update can be seen. The amount of pheromone added to the trails is determined by the solution quality shown in (4.6). It can be seen from the equation that cliques with higher number of vertices get more pheromone compared to other cliques. *m_vertices()* function in (4.6) gives the number of nodes in the maximum clique found so far.

$$\tau_{ij} \leftarrow \tau_{ij} + \left[\Delta\tau(ant_k) \cdot \frac{vertices(C_l)}{m\_vertices(ant_k)} \right], \quad l = 1..nbCliques \tag{4.6}$$

Algorithm 4 Weighted Global Pheromone Update

```
for each ant $k$ do
    for each clique $C_l$ do
        for each edge $ij$ do
            $\tau_{ij} \leftarrow \tau_{ij} + \left[ \Delta\tau(ant_k) \cdot weight \cdot \frac{vertices(C_l)}{m\_vertices(ant_k)} \right]$
        end for
    end for
end for
```

In the weighted version of global pheromone update, shown in Algorithm 4, the weights are used to differ selected ants when laying pheromones. This procedure is used in RAS to determine pheromone deposits, depending on the rank of the ant.

The local pheromone update algorithm, which is described in Algorithm 5, is only used in ACS and allows every ant to deposit its pheromones. The parameters ξ and τ_{initial}, where $0<\xi<1$ and τ_{initial} is the initial pheromone on the edge is used. Whether the ant creates the best solution or not, the pheromone is deposited on the edge for both feasible and infeasible solutions.

Algorithm 5 Local Pheromone Update

```
for each ant k do
    for each clique C_l do
        for each edge ij do
            $\tau_{ij} \leftarrow \tau_{ij} \cdot (1-\xi) + \tau_{initial} \cdot \xi$
        end for
    end for
end for
```

The differences in pheromone update procedures between ACO variants used in this thesis are described in the following subsections.

4.2.2.1 The pheromone update of ACS

In the ACS pheromone update process, shown in Algorithm 6, only the best-so-far ant is allowed to deposit its pheromones on the edges of its clique series. Evaporation is implemented at the same time of accumulation. Local pheromone update, shown in Algorithm 5, is still used for every ant.

Algorithm 6 ACS Pheromone Update

```
for each ant k do
    for each clique C_l do
        for each edge ij do
            $\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \rho \cdot \left[ \Delta\tau(ant_k) \cdot \frac{vertices(C_l)}{m\_vertices(ant_k)} \right]$
        end for
    end for
end for
```

4.2.2.2 The pheromone update of MMAS

In the MMAS pheromone update process, shown in Algorithm 7, the best-so-far, iteration-best ants are alternatively allowed to deposit pheromones on the edges of their clique series. Iteration-best ant is the best ant for a specific iteration, where best-so-far ant has the best solution of all iterations so far. There is one major difference between the original MMAS model and the model in our work: restarting for ants depending on a branching factor is not used in our solution. Thus, the diversity of solutions is not a problem and restart-best ant is not used in this process. In Algorithm 7, *u_gb* is used to select iteration-best and best-so-far ant alternatively throughout the algorithm. The value is chosen as 1, as restart-best ant is omitted.

Algorithm 7 MMAS Pheromone Update

```
if iteration % u_gb do
    Global Pheromone Update for iteration-best ant
else
    Global Pheromone Update for best-so-far ant
end if
```

4.2.2.3 The pheromone update of RAS

Algorithm 8 shows the pheromone update procedure of RAS. In RAS, a number of selected ants from a ranked list are allowed to deposit their pheromones, along with the best-so-far ant. The weight, wich will be used in the weighted global update procedure, is determined respectively to their ranks in the list. The best-so-far ant will have *w* value as weight, where r^th ranked ant will have *w-r* value as weight.

Algorithm 8 RAS Pheromone Update

```
for each ant $k$ that has rank $\leq w$ do
    Weighted Global Pheromone Update
end for
```

Overall flow for the update procedure is shown in Algorithm 9. Each ant finishes its trail, comes up with a solution and pheromone update procedure is processed afterwards.

Algorithm 9 Pheromone Update

```
for each ACO method do
    evaporate pheromones except for ACS
    call  the update procedure of the selected ACO method
end for
if MMAS then
    check pheromone limits on the trails
end if
for each ACO method do
    compute total pheromone as $\tau^{\alpha}_{ij} . \eta^{\beta}_{ij}$
end for
```

First, evaporation takes place on the edges with pheromones. Selected pheromone update procedure is run after evaporation step. Before the last step, if the ACO variant is MMAS, the pheromone limits on the edges are checked and corrected if there is an exceeded value. Pheromone trails are updated with the total information, defined as $\tau^{\alpha}_{ij} . \eta^{\beta}_{ij}$.

4.2.3 Solution Construction

The journey of each ant is limited with a tabu list; ants can not move to previously visited nodes. Ants will use the degrees of the nodes in the neighborhood of the current node as the heuristic information along with the pheromone trails on the edges. This information helps to choose the node with the maximum degree and the information becomes active depending on the pseudo-random proportional action choice rule. The construction steps are shown in Algorithm 10.

Algorithm 10 Solution Construction

```
for each ant $k$ do
    place ant on a random node
end for
while step < $n$-1 do
    step++
    for each ant $k$ do
        move to next eligible node
        if ACS then
            local acs pheromone update
        end if
    end for
end while
for each ant $k$ do
    pheromone trail update
end for
```

Ants can traverse all the nodes in the subgraph and if they get stuck along the journey (if they could not find any eligible node to add to thier trajectory), they are killed. Search continues until all ants are killed.

Selection of the next eligible node depends on the pheromone and heuristic information of the edge tied to that node. A probability ratio is used to determine dominance of pheromone and heuristic information; the selection decision is implemented with a pseudo-random proportional action choice rule. In this rule, shown in Algorithm 11, each eligible node is assigned a probability proportional value and ordered in an array. Cumulative probability value is calculated and the random node is selected after exceeding a defined probability parameter. The feasibilty of the node is defined by two parameters. If the node is unvisited and if the threshold value is not exceeded when the node is added, then the node can be selected for that ant to move. If there are more than one feasible candidate node, then the one with higher total information is chosen.

Algorithm 11 Solution Construction

```
prob_sum = 0
current_node = c
for each candidate node i do
    if not feasible then
        prob_ptr[i] = 0
    else
        prob_ptr[i] = total_information[c][i]
        prob_sum += prob_ptr[i]
    end if
    if prob_sum == 0 then
        choose best eligible node without total information
    else
        select a random node in prob_sum
        calculate the score of ant
    end if
end for
```

4.3 Fixing Overlapping Cliques

The resulting cliques, created by the best-so-far ants in each snowball piles, are naturally overlapping with at most one node. Traversing ants stop the constructed clique once they get stuck, and start a new one from the last node they are at. Eventually, a visited node in a previously created clique will be another clique’s initial node.

Fixing overlapping cliques is easy, as shared nodes between these cliques are detected. When two overlapping cliques are found, the shared node is added to the clique with the higher number of nodes and is deleted from the other. The details of this operation are explained in the first and second steps of Figure 4.1. Resulting cliques will have no shared nodes after this operation.

4.4 Transforming the Graph

After fixing the overlapping cliques, resulting cliques will be used to transform a new graph from the original one. The resulting non-overlapping cliques will be used as meta-nodes, called *clique-nodes*, to form a new graph which is smaller then the original. The number of edges and the number of nodes are reduced in the transforming process. With the reduced network graph, it will be possible to use a community detection algorithm in the next phase of the method.

The edges in the reduced graph will have weigths as they are actually merged edges into one edge for each node in the new graph; therefore, their weigth values should be re-calculated. The equation for the new edge weights is given in (4.7). Edge weight calculation is needed for the edges from clique-nodes to clique-nodes. The intra-community edge values of the cliques are also used in this equation.

$$e_{kl} = \frac{\sum_{i \in C_k} \sum_{j \in C_l} x_{ij}}{\min(\sum_{m \in C_k} \sum_{n \in C_k} y_{mn}, \sum_{p \in C_l} \sum_{r \in C_l} y_{pr})} \qquad \textbf{(4.7)}$$

In (4.7), x_{ij} is the relevance value of the edges between the clique-nodes while y_{mn} and y_{pr} represent the relevance values of the intra-community edges of the clique-nodes C_k and C_l. The edge weight e_{kl} of the newly formed edge between the clique-nodes or between a clique-node and an existing node is the result of the above equation. Higher values of edge weights mean a higher similarity.

In the ACO step, each thread provides a solution constructed from a set of quasi-cliques.After the overlap fixing step, the outputs of all threads are gathered and combined to find the whole set of quasi-cliques. The leftover nodes, which are not a member of any quasi-clique, are considered as nodes again in the new graph, while each quasi-clique is formed as clique-node. Then, we calculate the weights of the new edges in between the clique-nodes as well as the edges in between non-clique nodes with the created clique-nodes. The edges in between nodes which are not a member of any clique will also be preserved with their initial relevance values and are added to the new reduced graph.

4.5 Using a Community Detection Algorithm

The reduced graph with new edges and nodes (formed from original nodes and clique-nodes) is processed with a commonly used community detection method in the last step of the algorithm. Among the variants of community membership detection methods, the one which is able to process edge weights is chosen. The edge weights determine the strength of the relevance/similarity values between newly formed nodes in the reduced graph. The greedy method proposed in [7] is used to calculate modularity differences between the original graph and the reduced graph.

Clique overlapping steps (I and II) with graph transformation step (III) is shown in Figure 4.1. (I) and (II) illustrate the detection of a node shared by two cliques after which it is assigned to the clique with more nodes and removed from the other; (III) shows the resulting clique-nodes with the edge weight calculated between them as shown in (II).

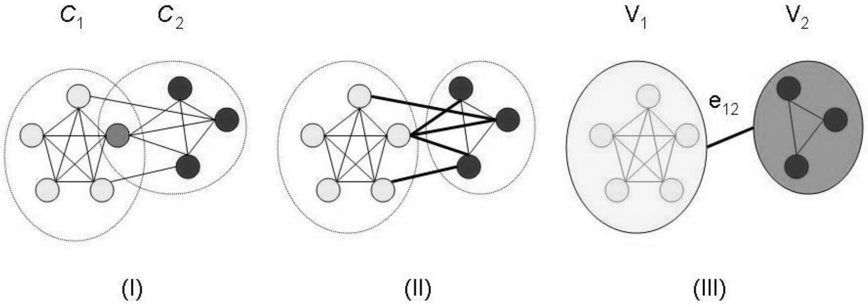


Figure 4.1 : Illustration of clique-node formation

5. EXPERIMENTAL STUDY

5.1 Experimental Setup

The proposed algorithm is coded in the C language and we used the iGraph [20] C library for the last step. For all of our experiments, we used a single PC (4GHz quad core processor with 16GBytes of main memory).

In the experiments, for each dataset, we first run iGraph on the whole (unreduced) graph and then on the graph reduced using our approach. We evaluate our results based on the number of communities detected, the node and the edge count reduction amounts, the Davies-Bouldin index values calculated as in [21] and the modularity values (Q) calculated using the iGraph community detection implementation.

5.1.1 Datasets

We used 9 network graphs for our experiments. The first 4 are the popular network grpahs used for benhcmarking in community detection area. The rest of the network graphs are used for performance testing and they are significantly large-scale in size. The information of the network graphs are given below, including the number of edge and node number of corresponding graphs. Following datasets are retrieved from [22,23,24].

5.1.1.1 Zachary's Karate Club

Social network of friendships between 34 members of a karate club at a US university in the 1970 [25]. The graph has 34 nodes and 78 edges. In the original data, 2 community groups are introduced.

5.1.1.2 Chesapeake Bay Food Web

A food web of lifeforms on Chesapeake Bay [26,27]. The graph consists of 34 nodes and 72 edges. In the original reports, 3 community groups are introduced.

5.1.1.3 Les Miserables

A social network graph for the characters played in the novel "Les Miserables" [28]. The edges show 2 character, each of which are represented with a node, are seen in the same scene. It consists of 77 characters and 254 connections.

5.1.1.4 American College Football

Network of American football games between Division IA colleges during regular season Fall 2000 [29]. The number of teams (nodes) is 115 and the number of matches played (edges) is 616. The reports say that there are 12 groups for the teams.

5.1.1.5 EPA

This graph was constructed by expanding a 200-page response set to a search engine query, as in the hub/authority algorithm [30]. The data is about the pages linking to www.epa.gov. It consists of 4,772 nodes and 8,695 edges.

5.1.1.6 Political Blogs

Political blogosphere Feb. 2005, compiled by Lada Adamic and Natalie Glance [31]. Links between blogs were automatically extracted from a crawl of the front page of the blog. It consists of 1,490 nodes and 19,090 edges.

5.1.1.7 Power Grid

An undirected, unweighted network representing the topology of the Western States Power Grid of the United States [32]. Data compiled by D. Watts and S. Strogatz. It consists of 4,941 nodes and 6,598 edges.

5.1.1.8 Free Online Dictionary of Computing

FOLDOC is a searchable dictionary of acronyms, jargon, programming languages, tools, architecture, operating systems, networking, theory, conventions, standards, mathematics, telecoms, electronics, institutions, companies, projects, products, history, in fact anything to do with computing [33,34]. The graph contains 13,356 words (edges) and 120,238 cross-references (edges).

5.1.1.9 Scientific Collaborations

Coauthorship network of scientists working on network theory and experiment, as compiled by M. Newman in May 2006 [35]. The graph contains 15,179 authors (nodes) and 79,934 coauthorship connections (edges).

5.1.2 Parameter Settings

Parameters which are specific to our algorithm and the ACO techniques used in our implementation are shown in this section, where *m* is the number of ants and *q0* is the psuedo-random proportional action choice parameter used in calculating the heuristic. The threshold value is used for quasi-cliques; it defines the acceptable unconnected node ratio for the next node to be added to the constructed clique. We performed initial experimentation to determine the ACO parameters given in the table that provided the best performance. In ACS we used ξ = 0.1 for the local pheromone update. For each of the datasets, we executed the algorithms 10 times with each run scheduled to complete in 15 seconds or 100 iterations, whichever occurs first. We used American College Football dataset and MMAS model, which provides better results compared to other variants, to tune our parameters.

First, we used the dataset to optimize the time based on the constructed cliques and achieved score for our proposed algorithm. The time is selected where the highest score with lower nuber of cliques, which means more successful cliques are constructed according to the proposal of the algorithm in Section 4. Figure 5.1 shows the effect of time interval to the number of cliques found and the achieved score. According to the figure, 15 seconds is enough to have optimum values as no valuable contribution is monitored beyond 15 seconds.

Time values are dependent on the CPU times of the computers we used in our experiments. According to that, the values in seconds may be changed in computers with different CPU speeds, however, the step counts will be similar to the graph we obtained. The seconds between 0-60 seconds are quantized to 12 samples; in faster CPU speeds, for instance max 30 seconds wil be quantized to 12 and the result will be same as the one we produced (i.e. 7.5 seconds time in faster CPU will be equal to the one we found on our computer will lower CPU speed with 15 seconds).

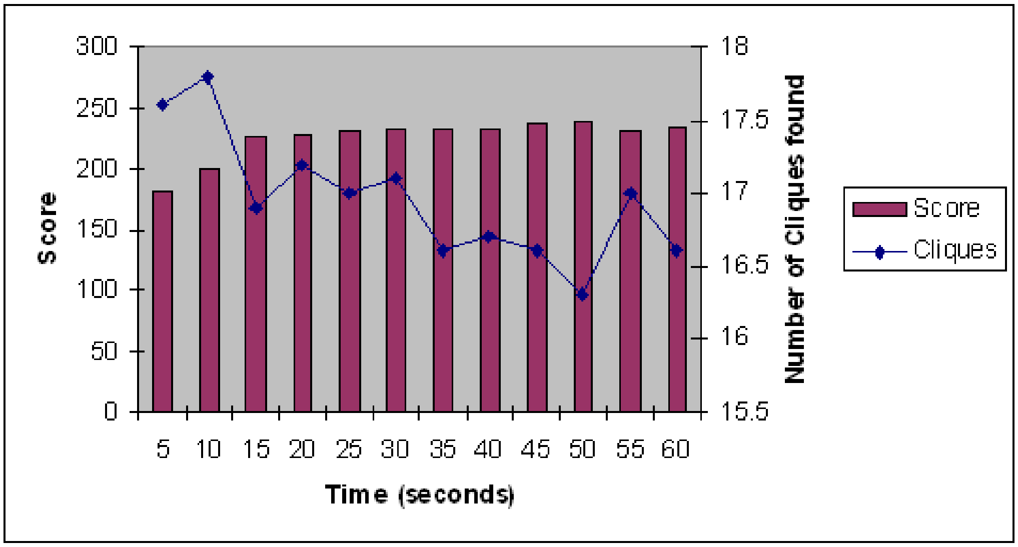


Figure 5.1 : Effect of maximum allowed time on cliques&score

Next, the effect of number of ants to the number of cliques and achieved score is investigated. In Figure 5.2, it is clearly seen that after number of ants is beyond 10 ants, the quality of the cliques found are decreased significantly, according to the degrading score. The reason behind this decrease is the amount of pheromones deposited on the edges. As the number of solution is limited by 100, which is also the iteration count given as the termination condition, each ant finds only 1 solution when the number of ants is given as 100. In that condition, the anst will not be able to use pheromone matrices as the termination condition is met. As an outcome, the amount of pheromone usage increases asthe number of ants decreases. The best results are achieved when the number of ants are chosen as 5, where the ants find 20 solution supported with the pheromone matrix.

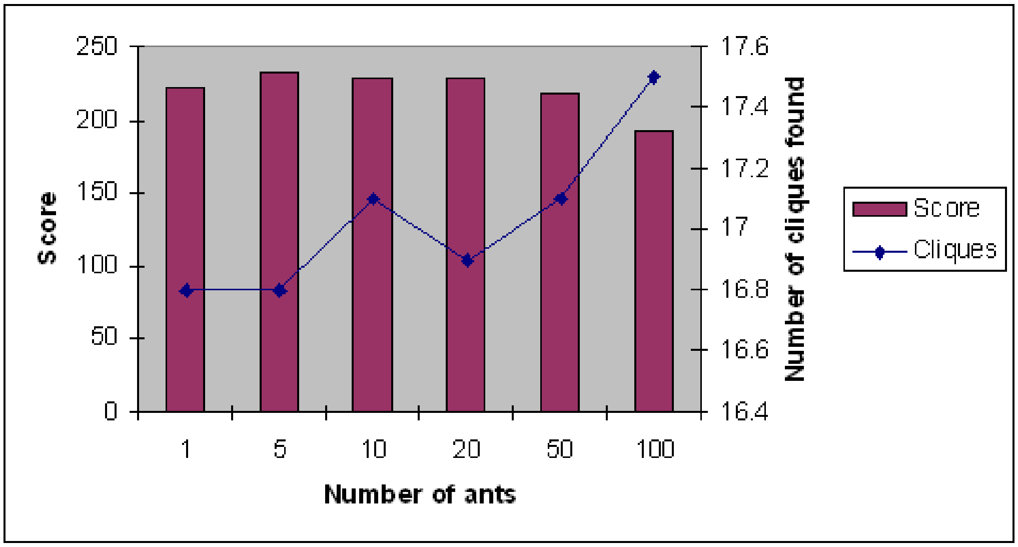


Figure 5.2 : Effect of number of ants used on cliques&score

The weights of pheromone information and the heuristic information used on the total pheromone matrix of the process are represented with α and β. α=0 means the choice of the node with highest heuristic information while β=0 means only pheromone information for the next node is used, which is explained on Section 3. The parameter tweaking is based on the values higher than 0 for those 2 parameters. In Table 5.1 and 5.2, the number of cliques found and the achieved scores are listed according to the varying values of α and β. The tests are done with the integer values of α and β, upto α=3 and β=4. The best results are obtained using α=2 and β=2.

Table 5.1: Effect of α and β on number of cliques found

α	β				
	0	1	2	3	4
0	17.8	17.1	17.2	17	17
1	16.7	17.2	16.8	17.4	16.7
2	16.9	16.3	16.4	16.7	17.1
3	17.4	17.1	17	16.8	17.2

Table 5.2: Effect of α and β on achieved score

α	β				
	0	1	2	3	4
0	187.47	185.27	184.23	192.79	200.54
1	231.32	227.97	230.09	227.83	230.12
2	229.28	230.28	236.61	234.50	230.31
3	231.50	230.47	232.15	232.14	234.36

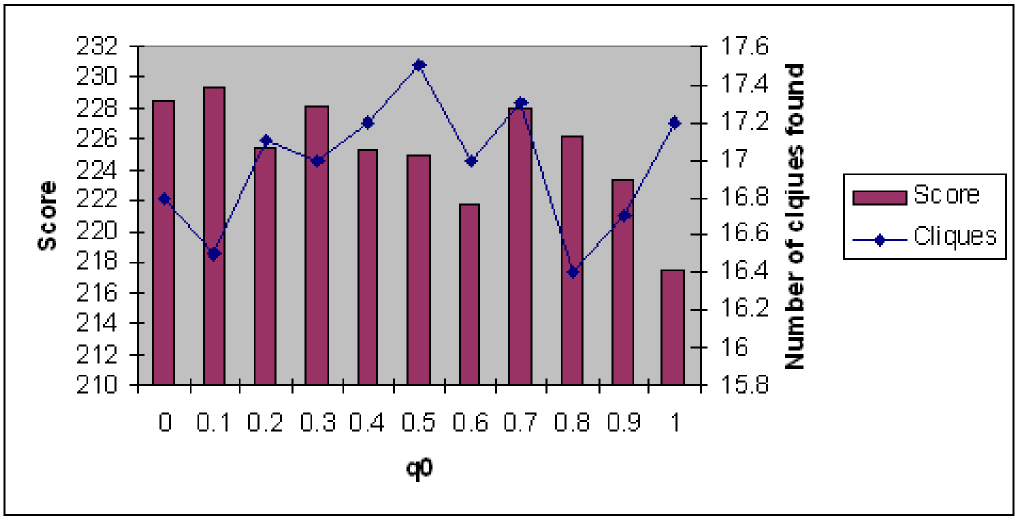


Figure 5.3 : Effect of q_0 on cliques&score

In the next step, the effect of q_0 is tested using the example dataset. The q_0 value is a probability threshold to determine the next move of the ants. For the values below the threshold value q_0, the edge with the best pheromone information is chosen, while for the rest of the values above q_0, which can be defined as 1-q_0, the pseudo-random proportional choice rule is applied. The details are covered in Section 3. The q_0 value brings randomness to the number of cliques found ans score achieved beyond 0.3. The best results in score is achieved on 0.1 and 0.7, however, 0.1 is selected as the number of cliques should be lower, which indicates better cliques are found. Higher values might provide diversity in the search space, however, it affected our proposed algorithm negatively.

The effect of ρ_0 is also tested. The results of the varying ρ_0 values are shown in Figure 5.4. According to the figure, ρ_0 is chosen as 0.1, where the best results are obtained.

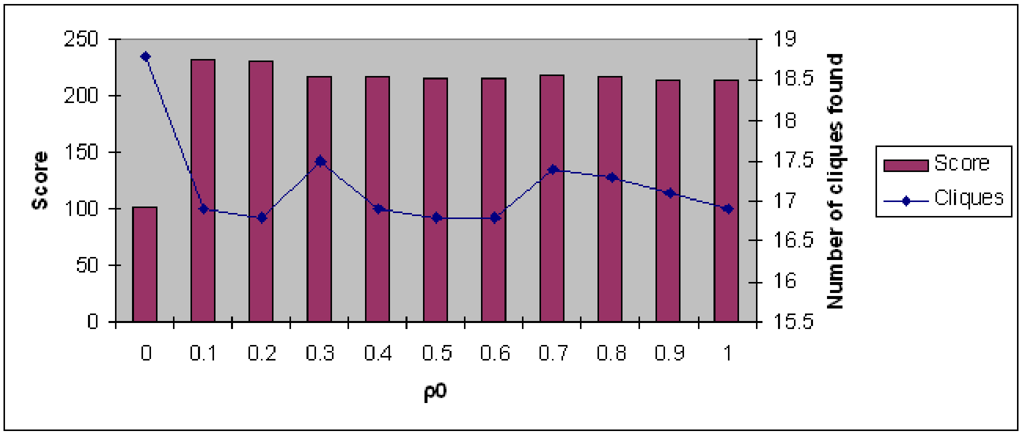


Figure 5.4 : Effect of ρ_0 on cliques&score

As the next step, the number of ranks used in RAS, shown as *w*, is determined. The results are shown in Figure 5.5. The maximum number of ranks is chosen as 2 according to the figure below.

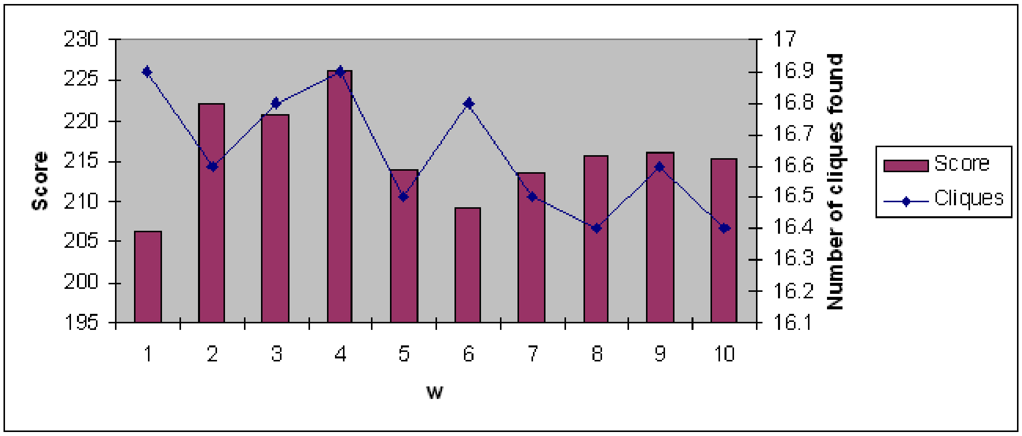


Figure 5.5 : Effect of *w* on cliques&score for RAS model

Branching factor is not used for MMAS approach. It actually brings diversity in the search space, however, as the ants traverse all the nodes in the search space by default, the branching factor is omitted for the MMAS model.

The number of threads used on datasets for subgraph sampling via Snowball Sampling method is determined manually according to the size (node+edge) of the corresponding dataset. As the size of the dataset grows, an appropriate number of threads are assigned to run on the dataset and divide the graph into that number of snowball samples. Following that rule, no multiple threads are run in smaller dataset. The number of snowball threads used in the process for each dataset is given in Table 5.3.

Table 5.3: Number of threads used on datasets

Datasets	Karate	Chesapeake Bay	Les Miserables	Football	EPA	PolBlogs	Power Grid	FolDoc	Scientific
Threads	1	1	3	5	10	10	10	20	20

In addition to our parameter optimization test run on Football data, we also tested our parameters on a bigger data for validation: Scientific Collaboration data. The results we obtained are nearly same with the results obtained on Football data, with some minor exceptions on q_0 and α/β tests. We will be using the parameters which we optimized on Football data, you can further analyze the results we found on Scientific Collaboration data in the Appendix section.

5.2 Experimental Results

We present the experimental results in Tables 5.4 to Table 5.11. Table 5.3 and 5.4 show the overall reduced cliques for datasets on each ACO model (ACS, MMAS, RAS). We run the ACO techniques for 10 times for each execution and calculate the results over 10 execution. The tables contain the best results of each ACO model, according to the threshold value used in the datasets.

With the lower-upper bounds of the confidence interval for number of reduced cliques with confidence level of 95% for the given datasets, there are no signifcant differences between ACS, MMAS and RAS along with the change on the threshold value. According to the relaxing of the threshold value, the number of cliques found per dataset with any of the ACO models decrease eventually. As the threshold value increases, the next node availability in the graph increases because required number of edges per node decreases and the cliques can grow bigger with the availability of new nodes. Another answer to insignificant difference between the ACO models on the number of cliques is the randomness of the starting point of clique generation. As a result of the random starting point of clique construction, possible bigger cliques can be scattered into many smaller cliques, thus resulting in more but smaller cliques found by the clique-finding ant. The results of the cliques found is more meaningful with the addition of achieved scores.

Table 5.4: Lower and upper bounds of number of cliques found with 95% confidence interval (part1)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
Karate	0	6.20	6.39	5.49	5.70	5.90	6.09
	0.1	6.29	6.50	5.60	5.79	6.11	6.28
	0.2	5.78	6.01	5.99	6.00	5.86	6.13
	0.3	5.49	5.70	5.29	5.50	4.83	4.96
	0.4	8.50	9.29	8.14	9.05	8.37	9.22
	0.5	6.68	7.31	5.64	6.35	5.58	6.41
Chesapeake	0	10.39	10.60	10.11	10.28	10.13	10.46
	0.1	10.20	10.39	10.39	10.60	10.16	10.43
	0.2	10.11	10.28	10.07	10.32	10.20	10.39
	0.3	9.98	10.21	10.43	10.76	9.90	10.09
	0.4	8.04	8.35	8.16	8.63	7.83	8.36
	0.5	7.83	7.96	7.04	7.35	7.33	7.66
Les Miserables	0	13.99	14.00	13.29	13.50	11.39	11.60
	0.1	10.90	11.09	12.16	12.43	13.16	13.43
	0.2	11.99	12.00	11.95	12.24	11.99	12.00
	0.3	11.90	12.09	12.99	13.00	12.03	12.16
	0.4	13.79	15.00	16.66	17.53	12.41	13.18
	0.5	10.64	10.95	12.84	13.35	11.42	12.17
Football	0	27.86	28.13	24.92	25.27	21.04	21.35
	0.1	25.11	25.48	22.98	23.21	28.25	28.74
	0.2	20.71	20.88	22.71	22.88	18.83	18.96
	0.3	18.90	19.09	19.67	19.92	20.60	20.79
	0.4	15.71	15.88	18.79	19.20	18.90	19.09
	0.5	19.04	19.35	18.78	19.01	17.52	18.07

Table 5.5: Lower and upper bounds of number of cliques found with 95% confidence interval (part2)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
EPA	0	504.43	504.96	506.76	507.43	522.32	523.67
	0.1	511.82	512.57	537.14	538.05	523.25	523.94
	0.2	517.36	518.03	506.03	506.76	522.65	523.34
	0.3	487.90	488.69	503.49	504.70	500.44	501.15
	0.4	1760.37	1762.42	1726.61	1729.18	1713.32	1714.87
	0.5	1290.12	1291.47	1291.94	1292.65	1288.20	1289.59
Political Blogs	0	229.34	230.25	242.59	243.60	230.05	231.14
	0.1	226.83	228.16	233.06	233.93	236.89	237.90
	0.2	223.14	224.05	218.05	219.14	218.39	220.00
	0.3	213.38	214.61	214.99	216.00	207.58	208.21
	0.4	333.50	347.09	368.36	372.43	364.16	370.43
	0.5	292.33	293.46	292.93	294.06	299.47	301.52
Power Grid	0	1606.12	1607.27	1595.09	1596.70	1598.43	1600.37
	0.1	1605.84	1606.95	1607.55	1608.84	1604.61	1606.38
	0.2	1609.76	1610.83	1599.28	1600.11	1607.08	1608.32
	0.3	1587.36	1588.63	1581.92	1583.07	1569.93	1571.66
	0.4	1345.13	1347.86	1361.85	1363.94	1350.75	1352.04
	0.5	1327.48	1329.31	1320.64	1322.95	1321.77	1323.82
FolDoc	0	3549.92	3551.47	3569.28	3571.71	3621.05	3622.94
	0.1	3545.48	3547.31	3542.14	3544.65	3545.20	3547.59
	0.2	3378.49	3381.10	3454.07	3456.92	3409.90	3411.29
	0.3	3251.35	3253.04	3215.25	3218.54	3262.19	3264.40
	0.4	3512.63	3515.76	3550.16	3554.03	3571.72	3574.27
	0.5	3121.97	3123.62	3117.67	3119.12	3078.59	3081.00
Scientific	0	3047.23	3048.76	3057.33	3059.06	3074.04	3075.95
	0.1	3045.68	3046.92	3073.13	3075.46	3070.16	3072.04
	0.2	3001.50	3002.69	3007.67	3009.52	3025.83	3028.16
	0.3	2952.59	2953.40	2952.68	2954.31	2966.86	2968.34
	0.4	3301.45	3302.74	3330.47	3331.52	3315.29	3317.90
	0.5	2932.60	2934.19	2931.46	2932.93	2932.68	2934.51

Table 5.5 and 5.6 show the achieved scores for datasets on each ACO model (ACS, MMAS, RAS). The lower and upper bounds of the achieved score with 95% confidence interval are given in the table for each defined threshold value. The threshold values vary from 0 to 0.5.

When we analyze Table 5.5 and 5.6, the results indicate no significant differences for smaller datasets with smaller values of threshold. While the results are relatively near, MMAS achieves little bit higher scores compared two other two. The reason for the corresponding good results on MMAS is the ability of MMAS to come up with better solution near to optimum values for short time intervals. The execution times of ACO trials grow in parallel as the size of the network dataset grows. ACS results in better solution with minor exceptions on some datasets. This is normal as ACS results in better solution for long time intervals in ACO trials. However, there is no significance difference to be named for ACS, MMAS and RAS for different datasets with different distributions. On the other side, if the threshold value increases, which results in the next node availability in the graph, the RAS model achieves better scores. The RAS model gives oppotunity to runner-up ants to lay their pheromones according to their solution quality, along with the best ant. This approach helps the model to balance the pheromone amount laid on the solution path, eventually preventing ants to be stuck at locally optimum values. Thus, when the availability of next node increase combine with RAS model, higher scores are achieved.

Table 5.6: Lower and upper bounds of achieved score with 95% confidence interval (part1)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
Karate	0	1011.62	1018.77	1078.84	1082.15	1049.19	1058.80
	0.1	1006.05	1013.14	1081.47	1085.52	1034.60	1052.39
	0.2	1177.88	1191.91	1202.70	1215.29	1144.27	1170.32
	0.3	1470.97	1499.22	1543.95	1564.04	1529.06	1546.53
	0.4	2351.30	2491.69	2296.63	2423.76	2401.71	2578.08
	0.5	4030.66	4376.53	3635.82	3950.37	4160.72	4407.47
Chesapeake	0	569.85	576.94	593.40	596.79	581.31	584.08
	0.1	577.75	580.04	588.45	592.94	581.80	584.59
	0.2	568.06	572.93	588.66	592.53	574.06	579.73
	0.3	674.50	680.69	674.09	676.30	661.76	666.43
	0.4	1753.54	1790.85	1805.09	1848.51	1793.99	1826.80
	0.5	2827.91	2945.88	2855.98	2938.81	2905.02	3009.97
Les Miserables	0	32538.16	32599.84	13796.27	13873.33	13389.47	13524.73
	0.1	47715.64	47749.76	10878.74	11207.26	12838.98	13388.42
	0.2	59956.07	60625.93	23754.17	23824.03	20653.18	21039.22
	0.3	17747.86	17924.54	13081.39	13163.61	40827.98	41019.82
	0.4	62100.30	62532.90	125007.10	125757.10	34206.17	34664.03
	0.5	102102.30	124672.10	127418.50	149901.10	50887.31	51190.49

Table 5.7: Lower and upper bounds of achieved score with 95% confidence interval (part2)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
Football	0	35110.20	35355.40	35563.38	36742.02	37655.78	38009.42
	0.1	21715.01	21834.19	42201.26	43092.54	26031.99	26175.21
	0.2	40095.42	40247.58	25531.72	25645.48	77739.50	79585.90
	0.3	54070.79	54790.81	50920.19	51399.01	72705.30	73445.50
	0.4	139679.80	141619.40	103123.80	103829.20	131397.20	133465.80
	0.5	111210.70	113352.30	91181.25	92636.35	128761.60	131682.20
EPA	0	4307.21	4308.38	4301.88	4302.91	4313.97	4315.42
	0.1	4337.03	4338.36	4313.70	4315.29	4356.73	4358.06
	0.2	4323.49	4324.70	4323.87	4325.12	4332.71	4333.88
	0.3	4409.66	4412.13	4426.73	4428.26	4413.12	4416.87
	0.4	10190.45	10202.95	10061.26	10079.74	10092.41	10105.39
	0.5	17913.25	17934.75	17826.08	17860.52	17887.17	17920.43
Political Blogs	0	13147.62	13804.18	12939.80	14153.20	12764.35	13402.65
	0.1	26579.08	27399.92	49747.81	50165.99	21282.45	22827.95
	0.2	35965.87	37397.73	28192.96	29069.04	21334.69	22381.31
	0.3	115322.40	118012.80	558767.40	563496.40	184024.90	190005.90
	0.4	318720.10	324982.90	781842.00	788547.60	220731.40	223620.80
	0.5	593288.70	603165.90	491413.00	498886.20	1144048.00	1159505.00
Power Grid	0	4790.30	4792.69	4610.53	4611.86	4720.18	4722.61
	0.1	4763.44	4767.75	4649.09	4650.70	4673.85	4674.94
	0.2	4658.56	4659.83	4651.30	4654.09	5453.37	5485.62
	0.3	4736.13	4739.47	4872.34	4875.26	4965.49	4970.70
	0.4	9920.90	9955.09	9928.39	9945.00	9958.55	9968.24
	0.5	15915.04	15944.76	15713.18	15729.42	16090.17	16118.83
FolDoc	0	16566.22	16618.38	16599.90	16619.70	16412.70	16446.70
	0.1	16207.62	16223.38	16133.58	16163.22	16368.92	16377.88
	0.2	19384.80	19444.80	19407.28	19446.72	19092.19	19126.21
	0.3	22025.27	22111.73	24589.99	24668.21	23941.37	24018.03
	0.4	40498.10	40626.30	40688.51	40999.69	40278.24	40540.16
	0.5	58242.92	58438.48	60981.79	61120.41	62540.87	62648.93
Scientific	0	43776.08	44018.52	54318.81	55085.59	45783.36	46026.04
	0.1	47305.46	47830.34	45318.65	45508.15	47752.06	47952.94
	0.2	47406.22	47614.78	48116.12	48377.28	44435.83	44861.77
	0.3	46889.79	47177.41	48492.21	48993.99	58608.97	59334.23
	0.4	92276.90	93149.10	70894.36	71443.44	74646.15	76009.45
	0.5	93313.46	93727.34	92569.69	92775.11	202292.20	202715.80

The lower and upper bounds of the number of communities found on the reduced graphs of datasets with 95% confidence interval along with the number of communities found on the original graphs are presented on Table 5.8, 5.9 and 5.10. The number of communities data is retrieved from the community detection tool used in our proposed algorithm. The results of the fast greedy method, explained in Section 4, provide the number of communities found in the original graph , the graph without preprocessing proposed in this thesis, and the reduced graph. The results in the table show that the ACO models, again, do not possess any significant difference on the number of comunities found. The difference on the community count is obtained with the change of threshold value. Once the threshold limit is incremented, the next availability increases. As a result, the size of constructed quasi-cliques also increases. This explains the construction of bigger clique-nodes and the decrease in the number of communities.

Table 5.8: Lower and upper bounds of number of communities found with 95% confidence interval (part1)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
Karate	0	3.99	4.00	3.99	4.00	3.99	4.00
	0.1	3.99	4.00	3.99	4.00	3.99	4.00
	0.2	3.71	3.88	3.99	4.00	3.99	4.00
	0.3	3.83	3.96	3.99	4.00	3.83	3.96
	0.4	2.83	2.96	2.49	2.70	2.90	3.09
	0.5	2.20	2.39	2.20	2.39	2.11	2.28
Chesapeake	0	3.39	3.60	2.99	3.00	2.99	3.00
	0.1	3.11	3.28	3.03	3.16	3.03	3.16
	0.2	3.11	3.28	2.71	2.88	3.11	3.28
	0.3	2.26	2.53	2.60	2.79	2.83	2.96
	0.4	3.03	3.16	3.11	3.28	2.83	2.96
	0.5	2.29	2.50	2.29	2.50	2.29	2.50
Les Miserables	0	5.99	6.00	5.99	6.00	6.71	6.88
	0.1	6.99	7.00	6.71	6.88	5.03	5.16
	0.2	4.99	5.00	5.53	5.86	5.56	5.83
	0.3	4.71	4.88	4.81	5.18	5.49	5.70
	0.4	5.39	5.60	3.98	4.21	5.67	5.92
	0.5	4.46	4.73	4.29	4.50	5.16	5.43
Football	0	5.81	6.18	6.71	6.88	6.83	6.96
	0.1	7.11	7.28	5.83	5.96	6.29	6.50
	0.2	6.60	6.79	7.43	7.76	5.78	6.01
	0.3	6.39	6.60	6.83	6.96	7.03	7.16
	0.4	5.98	6.21	5.49	5.70	5.75	6.04
	0.5	5.46	5.73	5.16	5.43	4.20	4.39

Table 5.9: Lower and upper bounds of number of communities found with 95% confidence interval (part2)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
EPA	0	536.39	537.20	533.27	534.12	535.80	536.39
	0.1	537.82	538.77	533.09	533.70	536.66	537.13
	0.2	534.39	535.20	536.70	537.29	534.94	535.45
	0.3	532.15	532.84	535.48	536.11	537.28	537.71
	0.4	534.18	534.61	536.55	537.44	534.23	534.96
	0.5	532.39	532.60	532.43	533.16	532.99	533.40
Political Blogs	0	274.92	275.27	272.90	273.09	275.09	275.50
	0.1	273.72	274.07	275.83	276.36	274.98	275.21
	0.2	275.56	275.83	276.16	276.63	274.20	274.39
	0.3	275.11	275.48	275.43	275.76	275.30	275.89
	0.4	272.38	272.81	275.09	275.90	274.97	275.42
	0.5	272.56	272.83	275.21	275.58	273.90	274.09
Power Grid	0	40.49	41.10	41.89	42.50	41.03	41.56
	0.1	38.99	39.40	39.33	39.86	42.03	42.56
	0.2	40.19	40.80	44.66	45.13	40.72	41.47
	0.3	40.12	40.87	40.11	40.68	41.55	42.04
	0.4	47.35	48.44	42.81	43.78	49.06	50.13
	0.5	47.16	47.83	49.60	50.79	47.49	48.70
FolDoc	0	45.33	45.86	44.73	45.46	44.79	45.60
	0.1	44.65	45.34	43.61	44.38	45.79	46.60
	0.2	45.43	45.96	42.42	42.97	44.20	44.79
	0.3	44.33	45.46	43.88	44.51	46.77	47.42
	0.4	31.77	32.42	32.14	33.05	33.66	34.53
	0.5	25.37	26.02	25.20	25.79	25.72	26.27
Scientific	0	2556.56	2557.83	2558.75	2560.24	2557.12	2558.47
	0.1	2548.62	2550.37	2555.09	2556.91	2556.19	2557.60
	0.2	2547.72	2549.47	2556.89	2558.91	2552.35	2553.64
	0.3	2551.58	2552.81	2555.62	2557.18	2554.72	2556.27
	0.4	2542.95	2544.04	2546.34	2548.25	2545.13	2546.26
	0.5	2540.01	2541.18	2546.95	2548.04	2536.23	2537.16

Table 5.10: Number of communities on the original graphs

Datasets	Karate	Chesapeake Bay	Les Miserables	Football	EPA	PolBlogs	Power Grid	FolDoc	Scientific
Communities	3	5	5	6	541	276	40	17	2574

In Table 5.11 and 5.12, the number of nodes and edges of the reduced graphs for each dataset, after being processd by ACO, is listed. The reduced nodes are the total of clique nodes plus the unassigned nodes and the reduced edges are the newly emerged edges with relevance/similarity weight obtained after the transformation phase. The original node and edge count for each dataset is also provided in the tables for comparison.

Table 5.11: Number of nodes and edges in the reduced graphs (part1)

Datasets	Original Graph		Threshold	ACS		MMAS		RAS	
	node	edge		node	edge	node	edge	node	edge
Karate	34	78	0	22	30	23	35	22	31
			0.1	22	30	23	35	22	31
			0.2	21	29	22	32	21	29
			0.3	19	26	20	26	19	27
			0.4	18	26	19	27	19	27
			0.5	17	19	18	20	18	18
Chesapeake	34	72	0	21	43	21	42	21	43
			0.1	21	43	21	43	21	44
			0.2	21	43	21	43	21	43
			0.3	21	41	21	42	21	39
			0.4	17	25	17	26	14	23
			0.5	14	18	14	20	14	20
Les Miserables	77	254	0	40	56	39	65	39	65
			0.1	40	74	39	55	37	59
			0.2	37	56	38	60	37	55
			0.3	37	48	36	57	37	49
			0.4	33	37	34	53	34	51
			0.5	33	39	32	32	34	47
Football	115	616	0	48	228	41	191	39	193
			0.1	37	181	40	205	45	222
			0.2	35	166	40	190	39	192
			0.3	33	168	37	183	29	134
			0.4	26	122	28	131	28	135
			0.5	27	136	30	145	25	137
EPA	4772	8695	0	4204	7890	4186	7886	4190	7886
			0.1	4180	7896	4187	7867	4206	7945
			0.2	4177	7888	4206	7919	4190	7882
			0.3	4161	7788	4149	7668	4144	7742
			0.4	2662	5819	2664	5809	2677	5836
			0.5	2060	5046	2083	5011	2093	5032

Table 5.12: Number of nodes and edges in the reduced graphs (part2)

Datasets	Original Graph		Threshold	ACS		MMAS		RAS	
	node	edge		node	edge	node	edge	node	edge
Political Blogs	1490	19090	0	1149	9287	1132	9173	1145	9345
			0.1	1135	8902	1132	8757	1151	9601
			0.2	1125	7905	1141	8204	1132	7719
			0.3	1073	6377	1091	6773	1095	7139
			0.4	904	6017	870	5755	889	5502
			0.5	684	4455	690	4396	736	4901
Power Grid	4941	6598	0	3119	4283	3122	4285	3127	4294
			0.1	3141	4305	3118	4272	3132	4290
			0.2	3131	4288	3122	4275	3109	4275
			0.3	3074	4195	3095	4210	3099	4221
			0.4	2592	3624	2587	3620	2607	3658
			0.5	2099	3014	2115	3043	2107	3031
FolDoc	13356	120238	0	7257	56094	7220	55998	7226	56099
			0.1	7278	55906	7234	55880	7282	55838
			0.2	6902	53552	6936	53567	6967	54008
			0.3	6434	50541	6378	49938	6377	50110
			0.4	5158	49029	5232	49501	5190	49520
			0.5	4278	47735	4317	47512	4277	47553
Scientific	15179	79934	0	8522	14017	8511	13965	8499	14037
			0.1	8484	14006	8474	14033	8464	13961
			0.2	8411	13532	8388	13287	8377	13508
			0.3	8147	12664	8113	12679	8140	12607
			0.4	7177	11637	7150	11414	7200	11538
			0.5	6344	10051	6340	10176	6383	10145

The 3 ACO models do not show any significant differences on reducement like in number of cliques, total score achieved and the number of communities found. The results are inversely proportional to the number of cliques found by the models on each threshold value. As the threshold value increases, the number of created cliques increase, evantually the number of reduced node and edge count decreases. The observation is meaningful as the size of the cliques (number of node and edges which constructs the clique) has direct effect on the reducement. The main aim of the thesis is achieved as the the reducement rate on nodes and edges is nearly 50% for all the datasets, increasing to 60% - 65% on thresholds with value of 0.5. The reducement directly decreases the computational costs of the community detection

method used, however, the solution quality loss should be discussed. The quality results of the experiments are mentioned and discussed on Section 5.3.

5.3 Discussion

Parallel to our expectations from the proposed algorithm to reduce the network graphs to maintainable sizes for benefit of community detection methods, we recommend 3 of the ACO models implemented in our approach. Our experiments show that all of these models can be used to reduce the nodes and edges in the graph to half of the original counts, thus improving the time complexity proportional to *O((E.V/4)log(V/2)).* 3 ACO models achieved relatively similar solutions on the number of cliques but the achieved scores differ according to the size of the datasets. For smaller datasets, which are run in short time intervals, MMAS achieves better scores. On the other, for larger datasets, as the time interval becomes longer, RAS performed better scores. Thus, the selection of the ACO model can be determined according to the size of the network graph datasets.

For the discussion part, the quality of the solution obtained after our proposed algortihm for graph reduction, we will compare the results obtained with the algorihtm and the original graph results based on 2 different quality metrics described in the following paragraphs. For all tables, please note that, the original graph values are the values obtained by running the community detection tool on the graphs without our algorithm for graph reduction used. With the given original values and the values gathered after graph reduction, we can easily compare the quality preservation after processing of our proposed algorithm.

The result of the experiments show enough success on graph reducement. However, the solution quality of the community detection phase should also be examined, as we expect minimum loss on solution quality. The quality of the community detection can be examined upon the modularity values of the corresponding graph upon clustering. The lower and upper bounds of modularity values of datasets with 95% confidence interval are given in Table 5.13 and 5.14. The modularity values on the original graph is given in Table 5.15.

Table 5.13: Lower and upper bounds of modularity with 95% confidence interval (part1)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
Karate	0	0.403668	0.410332	0.39804	0.40196	0.395844	0.400156
	0.1	0.399864	0.406136	0.39704	0.40096	0.38706	0.39294
	0.2	0.38608	0.39392	0.385804	0.386196	0.38808	0.39592
	0.3	0.396472	0.403528	0.399236	0.402764	0.391628	0.394372
	0.4	0.289536	0.322464	0.246968	0.283032	0.306044	0.329956
	0.5	0.09012	0.10188	0.076376	0.093624	0.079552	0.094448
Chesapeake	0	0.363594	0.370456	0.368484	0.374257	0.370424	0.37777
	0.1	0.359424	0.367278	0.375561	0.381441	0.373377	0.3807
	0.2	0.377898	0.384653	0.370549	0.376753	0.37596	0.382324
	0.3	0.365736	0.370234	0.356602	0.362689	0.349382	0.356945
	0.4	0.300855	0.328691	0.333466	0.361747	0.374658	0.4024
	0.5	0.297826	0.321366	0.323444	0.337916	0.320561	0.342002
Les Miserables	0	0.638811	0.639987	0.578865	0.5975	0.547841	0.560576
	0.1	0.442979	0.443501	0.417137	0.440845	0.46379	0.496151
	0.2	0.466709	0.475588	0.457867	0.469113	0.486531	0.501696
	0.3	0.392185	0.399825	0.312035	0.330312	0.48452	0.509144
	0.4	0.397604	0.430341	0.463968	0.483985	0.423805	0.43927
	0.5	0.387872	0.417188	0.395346	0.418262	0.480524	0.497963
Football	0	0.479047	0.487018	0.538556	0.543395	0.53432	0.536756
	0.1	0.514104	0.51786	0.5059	0.511917	0.512358	0.517921
	0.2	0.487301	0.490546	0.536632	0.538472	0.495765	0.502629
	0.3	0.497268	0.501619	0.527284	0.529606	0.52492	0.529472
	0.4	0.47927	0.487343	0.46103	0.469732	0.439355	0.444249
	0.5	0.398788	0.408032	0.44322	0.459421	0.388	0.395726
EPA	0	0.655148	0.656654	0.659214	0.660303	0.657543	0.658944
	0.1	0.650361	0.651513	0.649785	0.651283	0.650028	0.651821
	0.2	0.65506	0.656264	0.645407	0.646928	0.651573	0.653353
	0.3	0.653479	0.654301	0.648111	0.649215	0.652919	0.654783
	0.4	0.628884	0.630954	0.634913	0.637114	0.639949	0.641498
	0.5	0.613145	0.614351	0.621439	0.623293	0.62556	0.626871
Political Blogs	0	0.416568	0.417939	0.407844	0.410276	0.410532	0.411698
	0.1	0.410276	0.411071	0.414424	0.416258	0.409612	0.411369
	0.2	0.409352	0.41055	0.4071	0.409603	0.419938	0.420768
	0.3	0.405487	0.408619	0.402022	0.403066	0.388147	0.392349
	0.4	0.387313	0.388752	0.362308	0.365102	0.38478	0.386595
	0.5	0.377516	0.378623	0.375751	0.378204	0.385587	0.38666

Table 5.14: Lower and upper bounds of modularity with 95% confidence interval (part2)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
Power Grid	0	0.927824	0.928401	0.929622	0.93007	0.928828	0.929091
	0.1	0.92987	0.930274	0.929017	0.929306	0.928129	0.928598
	0.2	0.926943	0.927741	0.930796	0.931176	0.928429	0.928806
	0.3	0.928749	0.92912	0.928452	0.928927	0.927953	0.92849
	0.4	0.93058	0.930888	0.931072	0.931632	0.931706	0.932234
	0.5	0.926314	0.926817	0.925925	0.926492	0.927348	0.927843
FolDoc	0	0.574478	0.575031	0.574717	0.57535	0.572347	0.573078
	0.1	0.572312	0.57274	0.57223	0.57278	0.573881	0.574378
	0.2	0.572631	0.573155	0.569872	0.570548	0.571532	0.572041
	0.3	0.574742	0.575142	0.573252	0.574017	0.575475	0.575965
	0.4	0.477	0.477795	0.474694	0.475217	0.477279	0.47789
	0.5	0.418211	0.418969	0.410429	0.411059	0.41864	0.419673
Scientific	0	0.851548	0.852266	0.850099	0.850614	0.850301	0.850809
	0.1	0.852626	0.852891	0.85416	0.854822	0.852482	0.853119
	0.2	0.852017	0.852553	0.853703	0.85411	0.852203	0.852655
	0.3	0.854317	0.854733	0.845571	0.846211	0.850461	0.851144
	0.4	0.838334	0.839024	0.831376	0.831953	0.836184	0.83728
	0.5	0.820714	0.82139	0.823044	0.823349	0.826819	0.827587

Table 5.15: Modularity values on the original graphs

Datasets	Karate	Chesapeake Bay	Les Miserables	Football	EPA	PolBlogs	Power Grid	FolDoc	Scientific
Modularity	0.380671	0.410783	0.547220	0.549741	0.619769	0.427749	0.934977	0.397325	0.563875

As described in the previous sections, the modularity value can be used as a quailty measure to validate the results of a clustering/community detection. The results of modularity falls in an interval between -1 to 1. For a totally random clustering, the result will be closer to 0. For better clustering, the result will be close to 1, while the result decreases to -1 if the clustering is not good. Compared to the modularity values of the original graph, the 3 ACO models achieve almost the same quality with the original graph and even better results are achieved for some datasets. This observation means that the loss on solution quality upon reducement is at minimum.

When we further analyze the modularity results, we can see that the increasing threshold values affect the modularity values negatively. Higher threshold values allow bigger quasi-cliques to be constructed, however, this relexation may result in a false clustering as once a node is assigned to a clique-node, that node is forced to be in the same cluster with the other nodes in that clique-node. The decrease rate does not change for each ACO model but it differs on some network graph datasets.

The drastic decrease in some datasets, parallel to increase in threshold values, can be explained with the effect degree distribution of the network graph on the community detection method. As stated in [2], the social network graphs tend to possess a degree distribution which fits Power Law. Other random network graphs can possess Normal distribution or Poisson distribution. We should examine if the degree distribution of a network affects the modularity values. For instance, two different degree distribution of two network datasets are given in Figure 5.6 and 5.7.

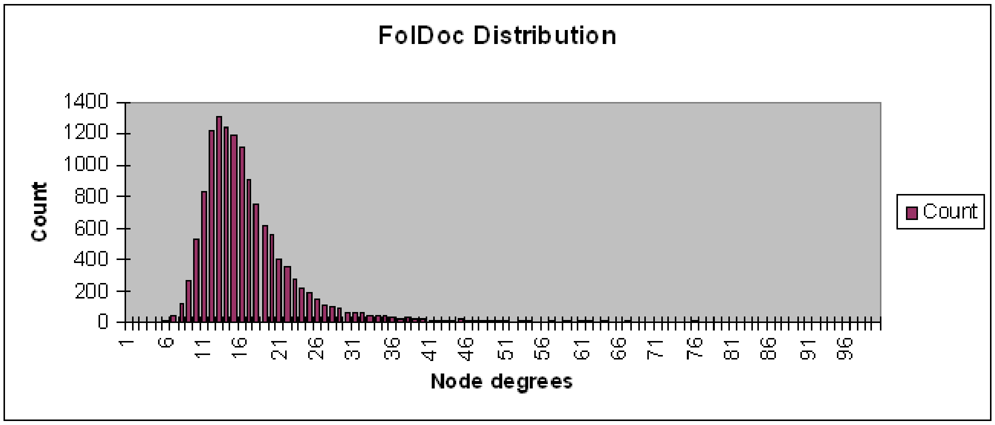


Figure 5.6 : Degree distribution of FolDoc dataset

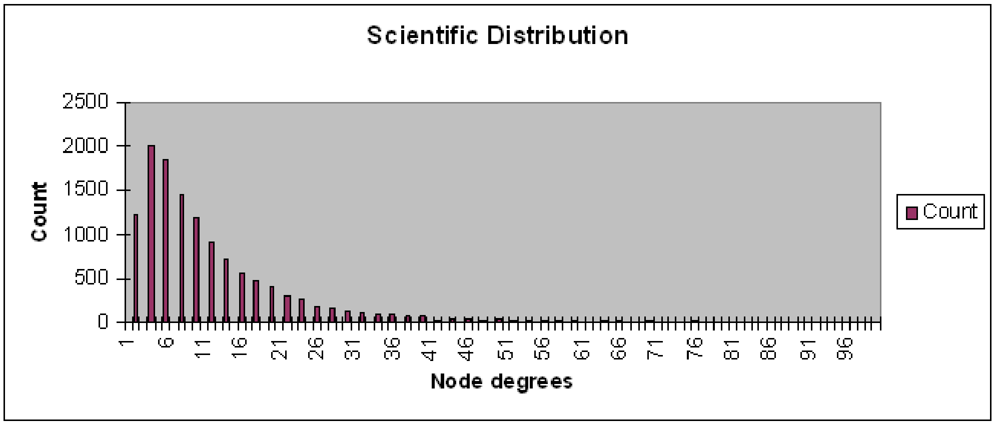


Figure 5.7 : Degree distribution of Scientific Collaboration dataset

With a quick observation from the figures of degree distribution, we can see that FolDoc data is formed in Poisson distribution while Scientific Collaboration network data nearly fits the Power Law. The modularity decrease in Scientific data is smaller compared to FolDoc, as the Power Law distribution allows more successful cliques to be found in the network graph and those cliques are more likely to be communities. In short, the increase in threshold allows bigger cliques and the decrease in the solution quailty is smaller as the better cliques are achieved on Power Law distributions. The decrease is smaller in graphs which fit Power law (i.e. Power Grid data) and bigger in graphs which do not (i.e. Les Miserables data). The rest of the degree distribution graps are provided in the Appendix section of this thesis.

Even though the results are satisfying, it is more appropriate to validate the results with another clustering validation metric, such as Davies-Bouldin Index [36]. The lower and upper bounds of Davies-Bouldin Index (DBI) with 95% confidence interval are given in Table 5.16 and 5.17, along with the Davies-Bouldin Index values for the original graph given in Table 5.18 for comparison.

Table 5.16: Lower and upper bounds of Davies-Bouldin Index with 95% confidence interval (part1)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
Karate	0	0.449864	0.456136	0.459804	0.460196	0.454452	0.459548
	0.1	0.44508	0.45292	0.457236	0.460764	0.462628	0.465372
	0.2	0.433472	0.440528	0.445804	0.446196	0.446668	0.453332
	0.3	0.439276	0.446724	0.4491	0.4589	0.45608	0.46392
	0.4	0.435256	0.440744	0.427572	0.444428	0.440492	0.449508
	0.5	0.390492	0.399508	0.337281	0.462719	0.387668	0.394332
Chesapeake	0	0.470387	0.476809	0.468414	0.473206	0.486313	0.489321
	0.1	0.488398	0.494249	0.47338	0.478062	0.47565	0.480205
	0.2	0.482527	0.486803	0.453152	0.465468	0.469201	0.47459
	0.3	0.420479	0.437776	0.441646	0.45661	0.444643	0.452867
	0.4	0.433023	0.448135	0.462936	0.477055	0.457877	0.47193
	0.5	0.429082	0.455745	0.420116	0.442693	0.425643	0.448803
Les Miserables	0	0.537748	0.53784	0.563839	0.564867	0.443641	0.449631
	0.1	0.467322	0.469814	0.473297	0.477897	0.569767	0.585481
	0.2	0.541829	0.542819	0.541324	0.54742	0.518085	0.536971
	0.3	0.568507	0.573933	0.49362	0.510295	0.513879	0.529239
	0.4	0.556303	0.568563	0.551825	0.566048	0.515349	0.525677
	0.5	0.479857	0.493807	0.496164	0.518421	0.419072	0.426501

Table 5.17: Lower and upper bounds of Davies-Bouldin Index with 95% confidence interval (part2)

Datasets	Threshold	ACS		MMAS		RAS	
		lower	upper	lower	upper	lower	upper
Football	0	0.438681	0.464885	0.416184	0.430706	0.387548	0.39866
	0.1	0.348668	0.362458	0.410034	0.431495	0.421287	0.438921
	0.2	0.408351	0.432444	0.333596	0.350448	0.441532	0.459489
	0.3	0.418327	0.436601	0.334979	0.352395	0.30346	0.307379
	0.4	0.470012	0.484645	0.450499	0.466039	0.415251	0.43945
	0.5	0.481567	0.50038	0.489562	0.516293	0.526652	0.534466
EPA	0	0.051359	0.052505	0.046064	0.047314	0.050398	0.051407
	0.1	0.052927	0.054306	0.046199	0.047188	0.05174	0.052449
	0.2	0.047981	0.049184	0.051557	0.052331	0.047727	0.048465
	0.3	0.044806	0.045972	0.049854	0.050883	0.052727	0.053528
	0.4	0.049518	0.050588	0.054355	0.055731	0.051571	0.052886
	0.5	0.046237	0.047073	0.046355	0.047145	0.046733	0.047839
Political Blogs	0	0.011909	0.012392	0.013099	0.013577	0.012886	0.013306
	0.1	0.011884	0.012345	0.015604	0.016819	0.01466	0.015685
	0.2	0.013049	0.013614	0.016836	0.017753	0.01154	0.012149
	0.3	0.016756	0.01777	0.027143	0.028113	0.027374	0.028901
	0.4	0.014334	0.015457	0.03167	0.033823	0.032682	0.034214
	0.5	0.0132	0.01394	0.025559	0.026961	0.022198	0.022821
Power Grid	0	0.943697	0.944923	0.94224	0.94384	0.943128	0.944857
	0.1	0.948375	0.94909	0.94058	0.943123	0.942102	0.943296
	0.2	0.938706	0.941796	0.944489	0.94573	0.940711	0.943312
	0.3	0.942101	0.943525	0.946115	0.948294	0.945443	0.946921
	0.4	0.927025	0.932236	0.942336	0.945531	0.925923	0.931505
	0.5	0.917491	0.919755	0.908852	0.911865	0.924913	0.927497
FolDoc	0	0.953799	0.954955	0.951363	0.953009	0.953062	0.954237
	0.1	0.950573	0.952453	0.952143	0.953235	0.952146	0.953362
	0.2	0.953492	0.95457	0.952801	0.953577	0.953284	0.954142
	0.3	0.955596	0.956183	0.954872	0.955497	0.952808	0.954076
	0.4	0.955086	0.955409	0.955326	0.95599	0.954981	0.955711
	0.5	0.949521	0.950153	0.950311	0.951073	0.9495	0.950608
Scientific	0	0.032967	0.033249	0.033096	0.03353	0.03222	0.032589
	0.1	0.030065	0.030587	0.03217	0.032599	0.031958	0.032338
	0.2	0.030278	0.030794	0.03282	0.033388	0.031868	0.032245
	0.3	0.031147	0.031508	0.032191	0.032606	0.031702	0.032102
	0.4	0.029452	0.029751	0.030185	0.030741	0.029821	0.030184
	0.5	0.028145	0.028506	0.030524	0.030854	0.027817	0.028142

Table 5.18: Davies-Bouldin Index values on the original graphs

Datasets	Karate	Chesapeake Bay	Les Miserables	Football	EPA	PolBlogs	Power Grid	FolDoc	Scientific
DBI	0.476318	0.450735	0.455000	0.505774	0.056011	0.010015	0.951465	0.912712	0.033027

There are many cluster validty metrics to be used, such as Silhouette validation, Dunn's Index and Davies-Bouldin Index [36]. In our experiments, we applied a modified version of the Davies-Bouldin index, introduced in [37]. The index depends on the values of the average diameter of the cluster and the average linkage between clusters. The index is actually the maximum value of averages of dissimilarity between a cluster with its most similar one, calculated with the aforementioned parameters.

The smaller values of Davies-Bouldin Index indicate better clustering for graphs by definition. The results of our experiments show that quality is preserved for each dataset while the threshold values change, compared to the original DBI values. In addition, the values are improving in some dataset, parallel to the threshold value incremental. In some datasets, the results seem to be significantly lower compared to other datasets. This can be explained with nodes without edges included for those datasets. As the number of nodes without edges increases, the DBI value closes to 0 as if there is a good clustering. These nodes are determined as 1-noded clusters by the community detection algorithm and it effects the DBI values in parallel. Following that, the higher number of communities found for those datasets, shown in Table 5.8, 5.9 and 5.10, can be explained with these unconnected nodes found on those datasets.

DBI values along with the modularities prove that the quailty is almost preserved for all dataset after our preprocessing. Thus, we recommend the usage of our algorithm for graph simplification.

6. CONCLUSION

Our main aim in this thesis is to reduce the size of the graph in order to be able to use community detection methods effectively on large-scale social network data sets. The quality of community detection methods is expected to be preserved while the original graph is reduced to a maintainable size. The preprocessing algorithm that we propose in this thesis for the community detection methods is an optimization problem from this perspective and its being resolved with a nature based approach.

There are many community detection methods developed through the start of researches on community detection on networks. The detection methods vary from hierarchical clustering to spectral bisectioning. The modularity maximization, namely the popular method for community detection, achieves good results in smaller time complexity, such as *O(E.VlogV)* after several modifications on the original algorithm. However, nearly all of these community detection methods suffer from high computational costs and non-scalability on large-scale social networks, even some of the methods are optimized to work on such networks. In this thesis, we are proposing a preprocessing methods for those community detection methods reduces the size of such networks without valauble information and solution quality loss.

The number of nodes and edges in the original graph is reduced through the concept of clique-nodes, which is implemented for our proposed algorithm. We used 3 ACO techniques (ACS, MMAS and RAS) to discover clique-nodes, which are then used to shrink the graph to a manageable size. The underlying motivation behind this approach is the fact that cliques are the base elements of communities. Based on the experimental results on various sized social networks, we may say that the execution times of the community detection methods are decreased while the overall quality of the solution is preserved. The results of the experiments show that the future optimization of the process may result in better solutions for community detection on very large-scale social networks.

For further optimization on performance of our preprocessing approach, there are some future directions which need to be pursued. For better parallelization, we will implement our solution on an existing parallelized framework. We will also pursue new parameters for ACO methods and investigate them (i.e. branching factor for MMAS). We will also further optimize our code to decrease our algorithm's own execution times to work more efficiently.

REFERENCES

[1] **Scott, J.,** *Social Network Analysis: A Handbook,* Sage Publications, London, 2002.

[2] **Girvan, M., and Newman, M.,** "Community structure in social and biological networks", *PNAS*, 99(12):7821-7826, June 2002.

[3] **Clauset, A., Newman, M., and Moore, C.,** "Finding community structure in very large networks", *Physical Review E*, 70(066111), 2004.

[4] **Donetti, L., and Munoz, M.,** "Detecting network communities: a new systematic and efficient algorithm", *Journal of Statistical Mechanics*, pp. 100-102, 2004.

[5] **Abello, J., Resende, M., and Sudarsky, S.,** "Massive quasi-clique detection", *5th Latin American Symposium on Theoretical Informatics*, pp. 598-612, 2002.

[6] **Fenet, S., and Solnon, C.,** "Searching for maximum cliques with ant colony optimization", *LNCS 2611*, pp. 236–245, 2003.

[7] **Wakita, K., and Tsurumi, T.,** "Finding community structure in mega-scale social networks", *16th International World Wide Web Conference*, Banff, Canada, May 2007.

[8] **Sadi, S., Uyar, A. Ş., and Öğüdücü, Ş.,** "Community Detection Using Ant Colony Optimization Techniques", *15th International Conference on Soft Computing, MENDEL 2009*, pp. 206-213, Brno, Czech Republic, 2009.

[9] **Goodman, L. A.,** Snowball sampling, *Annals of Mathematical Statistics 32*, pp.148–170, 1961.

[10] **Sadi, S., Uyar, A. Ş., and Öğüdücü, Ş.,** "An Efficient Community Detection Method using Parallel Clique-Finding Ants", *IEEE World Congress on Computational Intelligence (WCCI 2010),* Barcelona, Spain, July 2010.

[11] **Girvan, M., and Newman, M.,** "Finding and evaluating community structure in networks", *Physical Review E*, 69(026113), 2004.

[12] **Radicchi, F., Castellano, C., Ceconi, F., Loreto, V., and Parisi, D.,** "Defining and identifying communities in networks", *PNAS*, 101(9):2658-2663, March 2004.

[13] **Derenyi, I., Palla, G., and Vicsek, T.,** "Clique Percolation in Random Networks", *Physical Review Letters,* 94(160202), 2005.

[14] **Pizzuti, C.,** "Community detection in social networks with genetic algorithms", *GECCO'08*, Atlanta, Georgia, USA, July 2008.

[15] **Yan Liu, QingXian Wang, Qiang Wang, Qing Yao, and Yao Liu,** "Email community detection using artificial ant colony clustering", *LNCS 4537*, pp. 287–298, 2007.

[16] **Chevalier, C., Safro, I.,** "Comparison of coarsening schemes for multilevel graph partitioning"

[17] **Küçükpetek, S., Polat, F., and Oğuztüzün, H.,** "Multilevel graph partitioning: an evolutionary approach"

[18] **Dorigo, M.,** *Optimization, learning and natural algorithms*, PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 140 pages, 1992.

[19] **Dorigo, M., and Stutzle, T.,** *Ant Colony Optimization*, the MIT Press, 2004.

[20] **Csárdi, G., and Nepusz, T.,** "The igraph software package for complex network research", *InterJournal Complex Systems*, 1695, 2006.

[21] **Demir, G. N., Uyar, A. Ş., and Öğüdücü, Ş.,** "Multiobjective evolutionary clustering of Web user sessions: a case study in Web page recommendation", *Soft Computing - A Fusion of Foundations, Methodologies and Applications,* vol.14 no.6*,* pp. 579-597, Springer, Berlin, 2009.

[22] **Url-1,** Personal website developed by Mark Newman, Department of Physics, University of Michigan, http://www-personal.umich.edu/~mejn/, accessed May 2009.

[23] **Url-2,** Webpage developed by Vladimir Batagelj and Andrej Mrvar, <http://vlado.fmf.uni-lj.si/pub/networks/data/mix/mixed.htm>, accessed March 2010.

[24] **Url-3,** A network data repository website, University of California, <http://networkdata.ics.uci.edu/browse.php>, accessed April 2010.

[25] **Zachary, W. W.,** "An information flow model for conflict and fisson in small groups", *Journal of Anthropological Research* 33, pp. 452-473, 1977.

[26] **Baird, D., and Ulanowicz, R. E.,** "The seasonal dynamics of the Chesapeake Bay ecosystem", *Ecological Monographs* 59, pp. 329–364, 1989.

[27] **Url-4,** Courses webpage, developed by Mark Newman, Department of Physics, University of Michigan, <http://www-personal.umich.edu/~mejn/courses/2005/cscs535/index.html>,

[28] **Knuth, D. E.,** *The Stanford GraphBase: A Platform for Combinatorial Computing*, Addison-Wesley, Reading, MA, 1993.

[29] **Girvan, M., and Newman, M. E. J.,** *Proc. Natl. Acad. Sci. USA* 99, pp. 7821-7826, 2002.

[30] **Url-5,** Course webpage of Jon Kleinberg, *"The Structure of Information Networks"*, <http://www.cs.cornell.edu/courses/cs685/2002fa/>, accessed March 2010.

[31] **Adamic, L. A., and Glance, N.,** "The political blogosphere and the 2004 US Election", *in Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem,* 2005.

[32] **Watts, D. J., and Strogatz, S. H.,** *Nature* 393, pp. 440-442, 1998.

[33] **Howe, D.,** "*FOLDOC"*, <http://foldoc.org/>, 2002

[34] **Batagelj, V., and Mrvar, A.,** "*Pajek datasets"*, <http://vlado.fmf.uni-lj.si/pub/networks/data/>, 2006.

[35] **Newman, M. E. J.,** *Physical Review E*, 64(016131) & (016132), 2001.

[36] **Günter, S., and Bunke H.,** "Validation indices for graph clustering", *Pattern Recognit Lett* 24(8):1107–1113, 2003.

[37] **Speer, N., Spieth, C., and Zell, A.,** "Biological cluster validity indices based on the gene ontology", *Proceedings of advances in intelligent data analysis VI: 6th International Symposium on Intelligent Data Analysis (IDA 2005), LNCS 3646*, Springer, Berlin, pp. 429-439, 2005.

APPENDICES

APPENDIX A.1 : Parameter optimization results on Scientific Collaboration network graph data

APPENDIX A.2 : Degree distribution graph for whole datasets used in the experiments

APPENDIX A.1

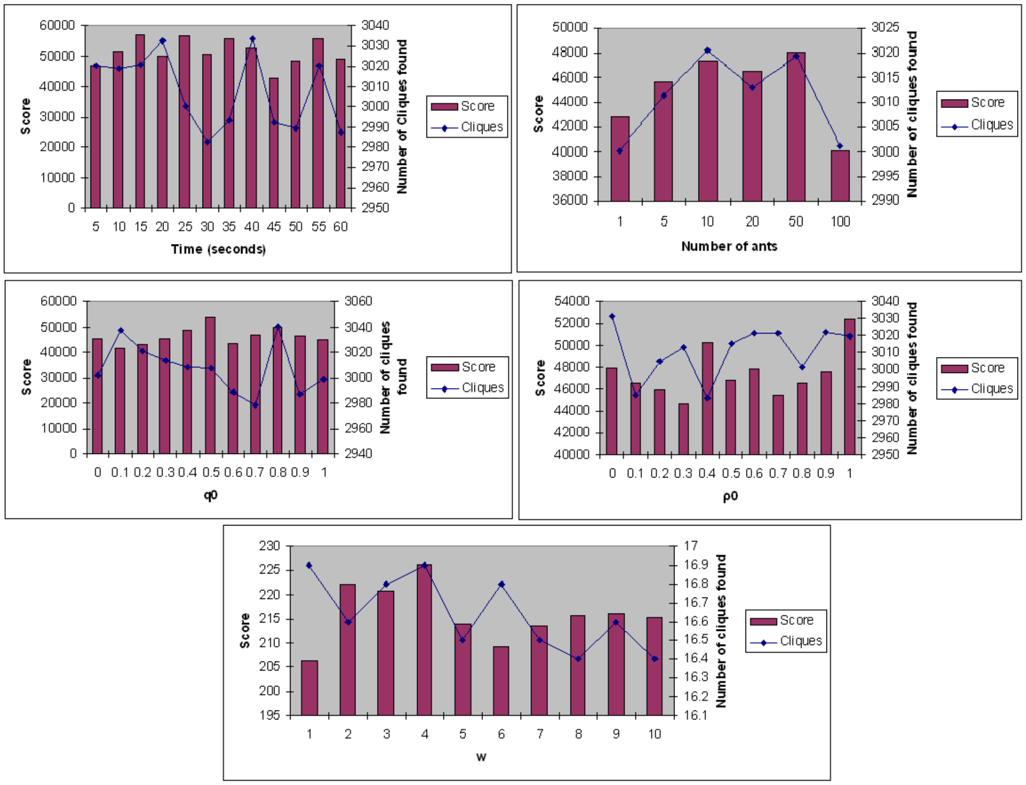


Figure A.1 : Parameter optimization results for Scientific Collaboration data

Table A.1 : Effect of *α* and *β* on number of cliques found

α	*β*				
	0	1	2	3	4
0	3047.1	3007.4	2991.8	3003.4	3010
1	3010.9	3001.1	3004	2998.7	2990.7
2	3003.8	2999.1	3003.7	3004.4	3014.2
3	3003.9	3024.5	3018.9	3013.7	2999.4

Table A.2 : Effect of *α* and *β* on achieved score

α	*β*				
	0	1	2	3	4
0	48210.9	48206.3	45379.3	42044.1	51975.5
1	45563.1	56008.9	48210.9	51377.9	43322.2
2	48388.8	46388.7	51619.9	42760.9	50010.7
3	45518.3	45205.6	46816.9	45449.8	48123.1

APPENDIX A.2

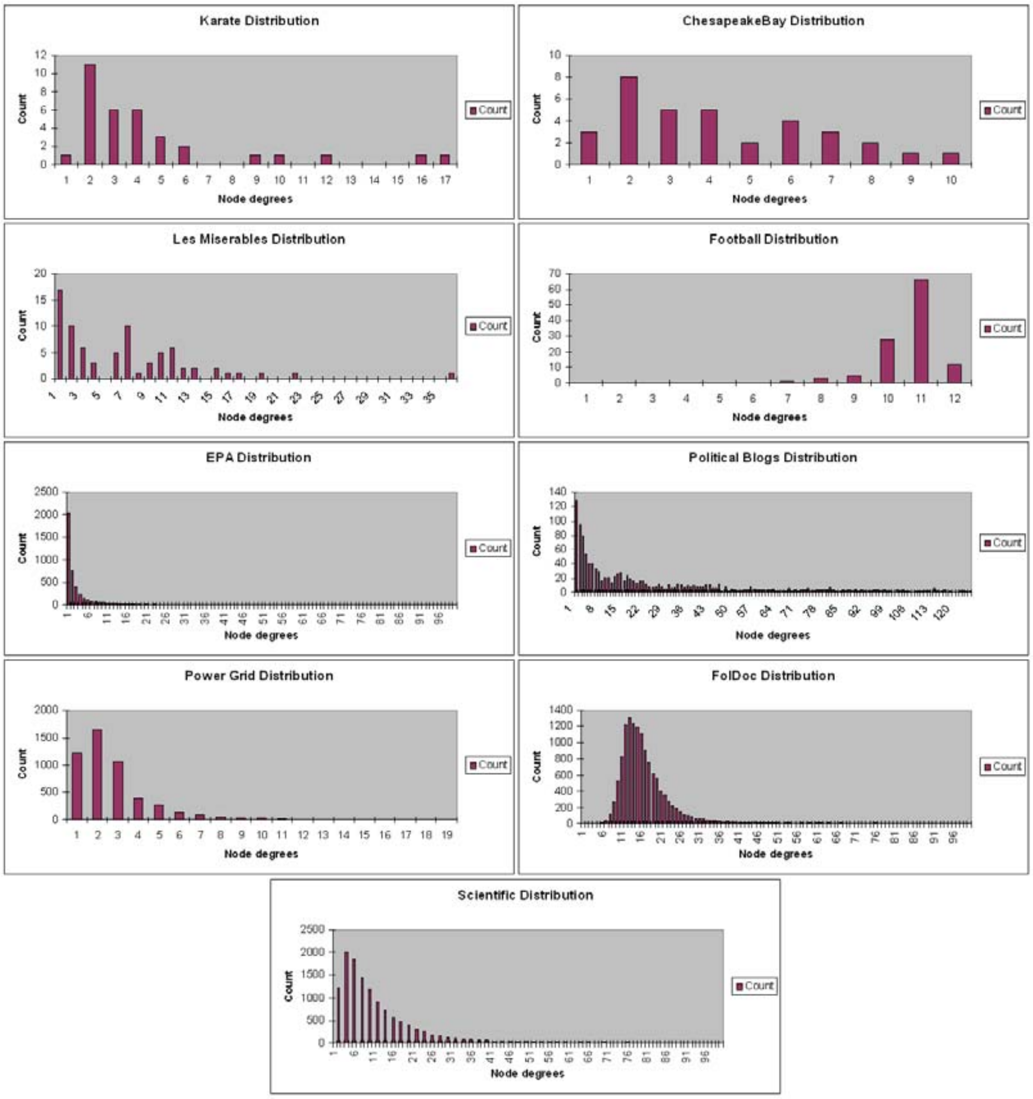
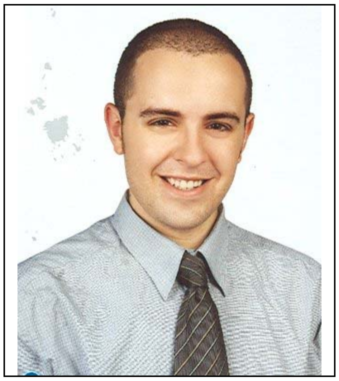


Figure A.2 : Degree distribution graphs for network datasets

CURRICULUM VITAE



Candidate's full name: Sercan SADİ

Place and date of birth: Istanbul/Turkey 13/08/1985

Permanent Address: Acarlar Sitesi G.Blok D.9 Ataşehir Istanbul/Turkey

Universities and Colleges attended: Computer Engineering, Istanbul Technical University
2003-2007

Ümraniye Anatolian High School
1997-2003

Publications:

- **Sadi, S., Uyar, A. Ş., and Öğüdücü, Ş.,** "Community Detection Using Ant Colony Optimization Techniques", *15th International Conference on Soft Computing, MENDEL 2009*, pp. 206-213, Brno, Czech Republic, June 2009.
- **Sadi, S., Uyar, A. Ş., and Öğüdücü, Ş.,** "An Efficient Community Detection Method using Parallel Clique-Finding Ants", *IEEE World Congress on Computational Intelligence (WCCI 2010),* Barcelona, Spain, July 2010.