Metaheuristic based peer rewiring for semantic overlay networks

Yulian Yang

Ordering N°2014ISAL0036

Thesis

Metaheuristic based Peer Rewiring for Semantic Overlay Networks

Application to Peer-to-Peer Information Retrieval

Submitted to

National Institute of Applied Sciences (INSA Lyon)
Doctoral School of InfoMaths

Università degli Studi di Milano (UNIMI)
Dept. of Computer Science

in fulfillment of the requirements for
DOCTORAL DEGREE

Prepared by Yulian YANG
Defended on March 28th, 2014

Examination Committee:

Rapporteur:

Massimo MELUCCI, Associate Professor	Università di Padova, Italy
Franck MORVAN, Professor	Université de Toulouse 3, France

Examinateur:

Catherine BERRUT, Professor	Université de Grenoble, France
Gabriele GIANINI, Assistant Professor	Università degli Studi di Milano, Italy
Marina RIBAUDO, Associate Professor	Università degli Studi di Genova, Italy

Directeur de thèse:

Ernesto DAMIANI, Professor	Università degli Studi di Milano, Italy
Lionel BRUNIE, Professor	INSA de Lyon, France
Sylvie CALABRETTO, Professor	INSA de Lyon, France

Acknowledgments

Pursuit of a PhD was an unexpected experience for me, and it turned out to be an amazing period in my life. It is those people, whom I met, knew, worked with, had too much fun with, and got a lot of help from, that made this period so precious; they made this thesis possible.

The depth of my gratitude goes to my three supervisors: Ernesto DAMIANI, Lionel BRUNIE and Sylvie CALABRETTO. Thank you so much for being flexible about what I did in this thesis, which allows me to explore the research domain according to my interests. Thank you for always being smiling and patient with me, which is the most powerful encouragement for a PhD student. Ernesto, thank you so much for the inspiring advices you gave about my work. They were so helpful and encouraging to pull me out of a lot of local optima. Lionel, thanks a lot for your efforts on my thesis manuscript. Without all of those comments and corrections, I would not be able to finish it, thank you! Sylvie, I want to thank you for all the supports you gave me during this period. Those supports to attend the academic events, to try to publish good papers, and to do reviews gave me the chances to communicate with more researchers in the world and be more open-minded.

I also own my gratitude to Gabriele GIANINI, with whom I've been collaborating. Since 2012, we started to work together on the modeling part of my thesis. The strong mathematical background of Gabriele made this work less difficult and more interesting. Thank you so much, Gabriele, for all of those discussions (sometimes disputes$\ddot{\smile}$).

My dear colleagues, from LIRIS, SESAR of Crema and the group of Distributed Information Systems of Passau, I own you so many thanks for all the joys we had and all the helps you gave me. Brother Guido, Tobias, and Lyes, thanks a lot for making the office home. Those coffee breaks, basketball sessions, biscuit times will be my most beautiful memory. Pierre-Edouard, you are the best person I met in France. Your personality and your kindness is one of the beautiful things I met during my PhD study. Sonia, David, Nadia, Stelvio, Vanessa, Romaric, Arezou and our dear secretary Réné, thank you so much for being nice to me and thanks for the helps you gave me. I also would like to thank all the organizers and participants of MDPS workshops. I feel lucky to be one of the members of this workshop. The presentations, discussions, and the social events of the workshops were one of the most exiting parts of my PhD life. Thanks you all for making this happen.

My dear friends, Francesca, Olga, Tarek and Salim, thanks for the happy time we have in Crema and in Lyon. Thanks for all the encouragement when I was in difficulty. The friendship we made and keep until now is an amazing achievement

of my life. Liangfen and Xiaolei, thanks for taking me as a member of the family. All of those greetings, lunches and dinners will be in my memory forever. To all the other of my friends, Guofeng, Zhen, Fangfang, Xinxin, Bing, Van Ahn, Yue, still too many names to mention, I am so glad to have met you in Lyon. Without you and our parties, life in Lyon would be less interesting.

Stefano, thank you so much for always being there, to encourage me and support me. Your positive attitude always cheered me up no matter how blue I was. Thank you so much for all the happy times we had together.

Finally, to my parents, my brothers and sisters, thank you so much for always being there. No words can express my gratitude to have you in my life. I love you.

Abstract: A Peer-to-Peer (P2P) platform is considered for collaborative Information Retrieval (IR). Each peer hosts a collection of text documents with subjects related to its owner's interests. Without a global indexing mechanism, peers locally index their documents, and provide the service to answer queries. A decentralized protocol is designed, enabling the peers to collaboratively forward queries from the initiator to the peers with relevant documents.

Semantic Overlay Network (SON) is one of the state-of-the-art solutions, where peers with semantically similar resources are clustered. IR can then be efficiently performed by forwarding queries to the relevant peer clusters in an informed way. SONs are built and maintained mainly via peer rewiring. Specifically, each peer periodically sends walkers to its neighborhood. The walkers walk along peer connections, aiming at discovering more similar peers to replace less similar neighbors of its initiator. The P2P network hence gradually evolves from a random overlay network to a SON.

Random and greedy walk can be applied individually or integrated in peer rewiring as a constant strategy during the progress of network evolution. However, the evolution of the network topology may affect their performance. For example, when peers are randomly connected with each other, random walk performs better than greedy walk for exploring similar peers. But as peer clusters gradually emerge in the network, a walker can explore more similar peers by following a greedy strategy. This thesis proposes an evolving walking strategy based on Simulated Annealing (SA), which evolves from a random walk to a greedy walk along the progress of network evolution. According to the simulation results, SA-based strategy outperforms current approaches, both in the efficiency to build a SON and the effectiveness of the subsequent IR.

This thesis contains several advancements with respect to the state-of-the-art in this field. First of all, we identify a generic peer rewiring pattern and formalize it as a three-step procedure. Our technique provides a consistent framework for peer rewiring, while allowing enough flexibility for the users/designers to specify its properties. Secondly, we formalize SON construction as a combinatorial optimization problem, with peer rewiring as its decentralized local search solution. Based on this model, we propose a novel SA-based approach to peer rewiring. Our approach is validated via an extensive experimental study on the effect of network rewiring on (i) SON building and (ii) IR in SONs.

Keywords: Peer-to-Peer Networks, Information Retrieval, Semantic Overlay Networks, Peer Rewiring, Local Search, Simulated Annealing

Résumé: Nous considérons une plate-forme pair-à-pair pour la Recherche d'Information (RI) collaborative. Chaque pair héberge une collection de documents textuels qui traitent de ses sujets d'intérêt. En l'absence d'un mécanisme d'indexation global, les pairs indexent localement leurs documents et s'associent pour fournir un service distribué de réponse à des requêtes. Notre objectif est de concevoir un protocole décentralisé qui permette aux pairs de collaborer afin de transmettre une requête depuis son émetteur jusqu'aux pairs en possession de documents pertinents.

Les réseaux logiques sémantiques (Semantic Overlay Networks, SON) représentent la solution de référence de l'état de l'art. Dans les SONs, les pairs qui possèdent des ressources sémantiques similaires sont regroupés en clusters. Les opérations de RI seront alors efficaces puisqu'une requête sera transmise aux clusters de pairs qui hébergent les ressources pertinentes. La plupart des approches actuelles consistent en une reconfiguration dynamique du réseau de pairs (peer rewiring). Pour ce faire, chaque pair exécute périodiquement un algorithme de marche aléatoire ou gloutonne sur le réseau pair-à-pair afin de renouveler les pairs de son cluster. Ainsi, un réseau à la structure initialement aléatoire évolue progressivement vers un réseau logique sémantique.

Jusqu'à présent, les approches existantes n'ont pas considéré que l'évolution de la topologie du réseau puisse influer sur les performances de l'algorithme de reconfiguration dynamique du réseau. Cependant, s'il est vrai que, pour une configuration initiale aléatoire des pairs, une marche aléatoire sera efficace pour découvrir les pairs similaires, lorsque des clusters commencent à émerger une approche gloutonne devient alors mieux adaptée. Ainsi, nous proposons une stratégie mixe qui applique un algorithme de recuit simulé (Simulated Annealing, SA) afin de faire évoluer une stratégie de marche aléatoire vers une stratégie gloutonne lors de la construction du SON. Les résultats de nos évaluations montrent que cette stratégie améliore les approches actuelles aussi bien pour la performance de la construction du SON que pour la pertinence des résultats retournés aux requêtes circulant sur le réseau pair-à-pair.

Cette thèse contient plusieurs avancées concernant l'état de l'art dans ce domaine. D'abbord, nous modélisions formellement la reconfiguration dynamique d'un réseau en un SON. Nous identifions un schéma générique pour la reconfiguration d'un réseau pair-à-pair, et après le formalisons en une procédure constituée de trois étapes. Ce framework cohérent offre à ses utilisateurs (i.e. concepteurs du réseau) de quoi le paramétrer. Ensuite, le problème de la construction d'un SON est modélisé sous la forme d'un problème d'optimisation combinatoire pour lequel les opérations de reconfiguration du réseau correspondent à la recherche

décentralisée d'une solution locale. Fondée sur ce modèle, une solution concrète à base de recuit simulé est proposée. Nous menons une étude expérimentale poussée sur la construction du SON et la RI sur SONs, et validions notre approche.

Mot-clés: Réseau Pair-à-Pair, Recherche d'Information, Réseaux Logiques Sémantiques, Recâblage des Pairs, Recherche Locale, Recuit Simulé

Abstract: L'oggetto dello studio è una piattaforma Peer-to-Peer (P2P) per Information Retrieval (IR).

Ogni peer ospita una collezione di documenti testuali con contenuti relativi agli interessi del suo proprietario. Senza l'utilizzo di un meccanismo di indicizzazione globale, i peer indicizzano localmente i loro documenti e forniscono il servizio di risposta a interrogazioni (query). La rete è dotata di un protocollo decentralizzato che rende possibile l'inoltro collaborativo delle query dal peer iniziatore ai peer con i documenti rilevanti.

Le Semantic Overlay Network (SON) sono una delle soluzioni allo stato dell'arte, dove i peer con risorse semanticamente simili sono raggruppati, formando un cluster. L'IR viene quindi realizzata efficientemente inoltrando le query ai cluster che si sanno essere rilevanti. Le SON sono costruite e mantenute principalmente attraverso l'operazione di peer rewiring. Nello specifico, ogni peer periodicamente invia un walker ai suoi vicini. Il walker segue le connessioni dei peer, con lo scopo di scoprire dei peer più simili al peer iniziatore di quanto non lo siano gli attuali vicini, per rimpiazzarli. La rete P2P network quindi evolve gradualmente da una rete casuale ad una SON.

Esplorazioni casuali e greedy possono essere applicate individualmente o in maniera integrata nel peer rewiring come una strategia generale durante l'evoluzione della rete. Tuttavia, l'evoluzione della topologia della rete può influenzare le prestazioni di queste modalità esplorative. Per esempio, quando i peer sono connessi casualmente l'esplorazione casuale opera meglio della esplorazione greedy per raggiungere peer simili, ma quando i cluster emergono gradualmente un walker può esplorare più peer simili utilizzando una strategia greedy. Questa tesi propone una strategia esplorativa basata su Simulated Annealing (SA), la quale evolve da una esplorazione casuale ad una di tipo greedy, seguendo l'evoluzione della topologia della rete. I risultati delle simulazioni dimostrano che la strategia basata su SA raggiunge migliori prestazioni rispetto agli approcci correntemente utilizzati, sia in termini di efficienza nella costruzione della SON, sia nell'efficacia della successiva IR.

Questa tesi propone diversi avanzamenti rispetto allo stato dell'arte in questo campo. Prima di tutto, si identifica un modello generico per il peer rewriting, formalizzato come una procedura in tre passi. La tecnica proposta fornisce una soluzione consistente per il peer rewiring, permettendo allo stesso tempo abbastanza flessibilità per gli utenti e i progettisti nella specifica delle proprietà del sistema. In secondo luogo, la costruzione della SON viene formalizzata come un problema di ottimizzazione conbinatorica, con il peer rewiring come strumento per la riceca locale della soluzione, in modalità decentralizzata. Sulla base di questo modello

viene proposto un innovativo approccio al peer rewiring basato su SA. Questo approccio è stato validato attraverso uno studio sperimentale estensivo degli effetti delle connessioni della rete su (i) la costruzione di SON e (ii) la IR nelle SON.

Keywords: Peer-to-Peer Networks, Information Retrieval, Semantic Overlay Networks, Peer Rewiring, Local Search, Simulated Annealing.

List of Figures

2.1 Categories of P2P overlay networks and P2P-IR, and the P2P network architectures and P2P-IR techniques employed by Semantic Overlay Networks (the thick arrowed lines) 20
2.2 Overview of SONs evaluation, the size of each section implies the amount of corresponding works 26

3.1 Repeated local search process of each peer (yellow part is performed by the walker to explore neighboring configuration; blue part is performed by the rewiring peer to update the current configuration with the explored ones). 44
3.2 An example of local search steps performed by a walker with TTL as 5, each peer is assumed to have 4 contacts (black arrowed lines: walker's path; blue circles: p_i's current neighbors; yellow circles: explored peers). 45

4.1 Probabilistic distribution of Metropolis and Glauber dynamics. . . . 57
4.2 Cooling schedule with initial temperature $T_0 = 500$. 67
4.3 Peers' dynamic behaviors can happen at any time during the network evolution, in which the clustering efficiency is assumed to increase monotonically in this illustration. 68

5.1 Optimization of relative intra-cluster similarity via random/greedy walk 82
5.2 Optimization of relative intra-cluster similarity with SA-based local search (random walk as the reference) 83
5.3 Cooling schedule with initial temperature 100 and different a 86
5.4 Analysis of peer rewiring behaviors: average distance between peers and their updated contacts 87
5.5 Optimization of clustering efficiency with random/greedy walk 88
5.6 Clustering efficiency with SA-based local search (random walk as the reference) 89
5.7 Evolution of the histogram of peers' clustering efficiency in different local search approaches (RC: rewiring cycles) 95
5.8 Evolution of IR recall in SONs generated via random/greedy walk . 96
5.9 Evolution of IR recall in SONs generated via SA-based local search (random walk as the reference) 98

5.10 Comparison of enhanced SA-based local search and SA-based local search in relative intra-cluster similarity 102
5.11 Comparison of enhanced SA-based local search and SA-based local search regarding clustering efficiency 102
5.12 Comparison of enhanced SA-based local search and SA-based local search in IR recall 103
5.13 Relative intra-cluster similarity of new peers (RC: rewiring cycles) . 105
5.14 Clustering efficiency of new peers (RC: rewiring cycles, $\gamma = 3$) 106
5.15 Relative intra-cluster similarity of SA-based local search with Glauber dynamics in different random network topology with 25000 peers (Random network topology 1 is the one we used to achieve the previous simulation results). 111
5.16 Clustering efficiency of SA-based local search with Glauber dynamics in different random network topology with 25000 peers (Random network 1 is the one we used to achieve the previous simulation results). 112
5.17 Relative intr-cluster similarity and clustering efficiency of SA-based local search with Glauber dynamics in random network topologies with different size (Network with 25000 peers is the one we used to achieve the previous simulation results). 113

List of Tables

2.1 Comparison of different P2P overlay networks 15
2.2 Typical works for P2P-IR 17
2.3 Overview of the techniques for generating and maintaining SONs . . 21

3.1 Example of a routing table 32

4.1 Exploring a peer p_i's neighborhood with SA 59
4.2 The probability of Metropolis dynamics given the value of the temperature and the energy difference 65
4.3 The probability of Glauber dynamics given the value of the temperature and the energy difference 65

5.1 Data sets used in state of the art (N/A: information is not available) 74
5.2 Parameters for the network configuration and SA-based local optimization 77
5.3 A summary of local search strategies 80
5.4 SA-based local search with Glauber dynamics: relative intra-cluster similarity/standard deviation 84
5.5 SA-based local search with Metropolis dynamics: relative intra-cluster similarity/standard deviation 85
5.6 Clustering efficiency/standard deviation of random and greedy walk (RC: rewiring cycles, $\gamma = 3$) 90
5.7 Clustering efficiency/standard deviation of SA-based local search with Glauber dynamics (RC: rewiring cycles, $\gamma = 3$) 91
5.8 Clustering efficiency/standard deviation of SA-based local search with Metropolis dynamics (RC: rewiring cycles $\gamma = 4$) 92
5.9 IR recall/standard deviation in SONs generated with 100 rewiring cycles via random walk and greedy walk 96
5.10 IR recall/standard deviation in SONs generated with 100 rewiring cycles via SA-based local search with Glauber dynamics ($k_q = 4$) . . 100
5.11 IR recall/standard deviation in SONs generated with 100 rewiring cycles via SA-based local search with Metropolis dynamics ($k_q = 4$) . 101
5.12 IR recall of s and θ with $N = 25000, l = 15, k_c = 7, k_q = 3$ 107
5.13 IR recall of s and k_q with $N = 25000, l = 15, k_c = 7, \theta = 0.5$ 108
5.14 IR recall of l and k_c with $N = 25000, s = 15, k_q = 4, \theta = 0.5$ 108

5.15 Number of strongly connected components with different values of N and l 109

Contents

1 Introduction **1**

1.1 Peer-to-Peer Information Retrieval . . . 2
1.1.1 Reference Scenario . . . 2
1.2 Problem Statement . . . 3
1.3 Methodology . . . 4
1.4 Contributions . . . 6
1.4.1 Generic Models for Building SONs . . . 6
1.4.2 SA-based Decentralized Local Search Solution . . . 6
1.4.3 Extensive Experimental Study . . . 7
1.5 Organization of the Thesis . . . 7

2 State of the Art **9**

2.1 Introduction . . . 9
2.2 Clarifying the Concepts . . . 10
2.3 P2P Overlay Networks . . . 12
2.4 P2P Information Retrieval . . . 16
2.4.1 Global Indexing . . . 17
2.4.2 Local Indexing . . . 18
2.5 Semantic Overlay Networks (SONs) . . . 19
2.5.1 SONs in Unstructured P2P Networks . . . 20
2.5.2 Evaluation of SONs . . . 25
2.6 Summary . . . 27

3 Optimization Model for Building SONs **29**

3.1 Methodology . . . 30
3.1.1 Similarity Measurement . . . 30
3.1.2 Definition of Links . . . 31
3.1.3 Building SONs via Peer Rewiring . . . 33
3.1.4 Searching in SONs . . . 37
3.2 Building SONs as an Optimization Model . . . 39
3.2.1 Building SONs: Combinatorial Optimization Problem . . . 39
3.2.2 Peer Rewiring: Decentralized Local Search Solution . . . 41
3.2.3 Discussion: From Decentralized Local Search to Global Optimization . . . 47
3.2.4 Related Work . . . 48

3.3 Summary 49

4 Simulated Annealing based Local Search for SONs 51
4.1 Introduction 51
4.2 Local Search: an Overview 52
4.3 Motivation 53
4.4 Simulated Annealing (SA) 55
4.5 SA-based Decentralized Local Search 58
4.5.1 Applying SA to Decentralized Local Search 58
4.5.2 Enhanced SA-based Decentralized Local Search 62
4.5.3 Cooling Schedule 64
4.5.4 Cooling Schedule with Peers' Dynamic Behaviors 67
4.6 Summary 69

5 Evaluation 71
5.1 Data Preparation 72
5.1.1 Data Requirement 72
5.1.2 Related Work 72
5.1.3 Using Reuters Corpus 75
5.1.4 Generating Peer Profile 75
5.2 Experimental Setup 76
5.2.1 Configuring the Network: Setting Parameters 76
5.2.2 Simulation Procedure 78
5.3 Metrics 80
5.4 Results and Analysis 81
5.4.1 Building SONs from Random Networks 81
5.4.2 Information Retrieval in SONs 94
5.4.3 Enhanced SA-based Local Search 99
5.4.4 Building SONs with Dynamic Behaviors: Peer Joining 103
5.4.5 Configuration VS. Performance 104
5.5 Summary 114

6 Conclusions and Future Work 117
6.1 Conclusions 117
6.2 Future Work 119
6.2.1 Designing Adaptive Cooling Schedule 119
6.2.2 Refining Peer Profile and Similarity Measurement 120
6.2.3 Refining Neighborhood Exploration 121
6.2.4 Improving IR Performance within Peer Clusters 121

Bibliography 123

Acronyms 135

Notations 137

Publications 141

Chapter 1

Introduction

Contents

1.1 Peer-to-Peer Information Retrieval 2
1.1.1 Reference Scenario 2
1.2 Problem Statement 3
1.3 Methodology 4
1.4 Contributions 6
1.4.1 Generic Models for Building SONs 6
1.4.2 SA-based Decentralized Local Search Solution 6
1.4.3 Extensive Experimental Study 7
1.5 Organization of the Thesis 7

A Peer-to-Peer (P2P) network is a decentralized distributed network architecture in which individual nodes (called *peers*) act as both suppliers and consumers of resources. It is in contrast to the centralized Client-Server (C/S) model where client nodes request resources provided by central servers.

A large number of P2P based systems have been developed for content (e.g., document) sharing, delivering and searching in last decades, such as P2P file sharing systems (e.g., [BitTorrent 2013]), P2P web search (e.g., [YaCy 2011]), and P2P social networks (e.g., [PeerSoN 2012]). With the content distributed in peers, all of these systems must provide a basic and necessary searching service, which is defined as Peer-to-Peer Information Retrieval (P2P-IR). It involves forwarding the information requests from their initiators to the peers having the relevant contents, and then getting the relevant contents back to the initiators. When an information request is issued by one peer, how to efficiently get the relevant content in other peers is still an open question.

One paradigm to perform P2P-IR is to build a Semantic Overlay Network (SON), an overlay network where peers with semantically similar content are clustered together [Crespo 2002b]. Queries can be forwarded to relevant peer clusters in an informed way instead of being blindly flooded over the whole network. Moreover, range queries or other advanced queries can be allowed in SONs [Doulkeridis 2010].

The choice of the specific paradigm to build SONs partially depends on the underlying network infrastructure. This choice in turn affects the robustness of the network against peers' dynamics behaviors (join/leaving the network, updating the content), the autonomy of the peers over their content, and even the performance of P2P-IR [Raftopoulou 2009a]. In general, similar peers can be clustered in structured P2P overlay networks, super-peer based P2P overlay networks, or unstructured P2P overlay networks. Among them, unstructured P2P overlay networks have the advantages of allowing high robustness to the dynamics and high autonomy of the peers. In this unstructured infrastructure, peers randomly connect to a limited number of other peers as their neighbors. They can leave or join the network without causing much workload to the network. However, peers can only communicate with their neighbors, which constitute the only knowledge they have about the network. Also, no central server exists to control the whole network and globally manage the resources in the network. All these factors make the clustering of similar peers to form a SON a challenging task, especially when the size of the network is large, and the peers frequently leave/join the network or update their contents.

This thesis considers an application of P2P-IR in which peers have full autonomy over their contents and can freely join/leave the network. Therefore, we focus on building SONs in unstructured P2P networks.

1.1 Peer-to-Peer Information Retrieval

1.1.1 Reference Scenario

Let's consider a decentralized system for sharing personal expertise among researchers. Each researcher has a collection of text documents about his/her speciality (could be more than one speciality), and keeps them in a personal computer or privately somewhere in a server (e.g., cloud). An software agent is used to manage the documents (e.g., indexing and querying) for each researcher. It has full autonomy over these documents (e.g., which file to share, which search technique for indexing and querying).

A large number of software agents self-organize into a P2P network—SON—where software agents are called peers. Each peer has a limited number of connections to other peers which are called the peer's *neighbors*. By keeping a connection to a peer, it means the keeper can directly send messages to the peer, although the messages may go through several routers in the physical network. The connections are directed: peer A connecting to peer B does not necessarily mean that peer B connects to peer A. They are self-organized by the peers themselves in such a way that peers with similar contents are connected and thus clustered, so that these

peers can access each other within few hops; the peer clusters are connected so that message can be forwarded from one cluster to another. With the self-organized SON, researchers can ask relevant information from other researchers by sending a request to the network. The request is post-processed by the software agent into a *query*. *Query routing* is then performed in the following ways: one or more relevant peer clusters are firstly identified according to the connections among peer clusters. Then the query is forwarded to the relevant cluster(s). Once a query reaches a relevant cluster, it is diffused among the peers in the cluster via the connections that cluster them up. These peers perform local IR and then return the relevant documents to the query initiator.

Besides the full autonomy over their documents collections, peers can autonomously join/leave the network or update their contents. The self-organized SON has a mechanism to efficiently restore the SON topology changed by these *dynamic behaviors*, so that keep it robust against these behaviors. Therefore the performance of the target tasks (e.g., IR) can not be affected.

The same P2P-IR scenario is also suitable to interest-based P2P social network, where users host web pages or documents which are related to their interests. By sending a query to the relevant user clusters in which users share the similar interest, the query initiator can get relevant information. Self-organized SON can also be applied to P2P Information Filtering, in which documents are recommended among peers sharing similar interest.

1.2 Problem Statement

The problem is how to build the SON given a randomly connected P2P overlay network and certain requirements and constraints.

Let's represent a self-organized P2P overlay network as $G = \langle P, L \rangle$ with P representing all the peers and L representing all the connections. Each peer p_i has a collection of text documents D_i in one/multiple themes, and a limited number of connections to the other peers (a limited number of neighbors). Initially, the neighbors of each peer are randomly sampled from the network. The connections must be rewired to the appropriate peers, in order to build SON that has the following properties:

Clustered: Each peer has a set of connections to the peers with similar document collections. Through these connections, peers are clustered, in such a way that each peer can access all the similar peers in the network by one or few hops. This property is used for diffusing information such as queries within a cluster.

Connected: Each peer has a set of connections to the peers with different document collections. These connections make the whole network connected. The property is used for identifying the peer clusters that are relevant to a query.

The following hypotheses of operational constraints are made according to the reference scenario:

No coordinators: Neither a central point nor powerful and stable peers exist to facilitate the SON construction;

Peer autonomy: Peers have full autonomy over their documents. They locally manage and index the documents they would like to share. Other peers can not access these contents unless they send a query to the peers to request relevant documents;

Limited connections: Each peer only keeps a limited number of connections to the other peers, considering that maintaining the connections consumes the resources of the peer as well as the networking facility.

In addition, in order to maintain its topology when dynamic behaviors like joining/leaving the network or updating contents happen, the SON must be able to perform the following operations:

Joining new peers: new connections must be added in the network in order to associate the new peers with the other similar peers in the network;

Recovering broken connections: old connections must be rewired when one of the connected peer leave the network and break the connection;

Rewiring changed connections: old connections must be rewired, when one of the connected peer updates its content and thus change their similarity.

With the above constraints of local operations and network dynamics, it is challenging to design a protocol to build a SON efficiently. By efficiently, it means to build a SON with a high quality of peer clusters and with a low cost of time and traffic.

1.3 Methodology

Given a random P2P overlay network as previously described, the SON is built via *peer rewiring*: peers rewire their connections so that each peer has a set of connections to peers with similar contents (called *short-range links*) and a set of

connections to peers with different contents (called *long-range links*). Similar peers are then clustered up by short-range links and peer clusters are connected by long-range links. Long-range links can be easily obtained, since there are usually much more peers with different contents than peers with similar contents. So the rewiring mainly focuses on building short-range links.

We identify and formalize peer rewiring as a three-step procedure repeatedly performed by each peer that initially connects to some random peers. A peer that rewires its connections is called *rewiring peer*. The three-step procedure is described as follows:

Rewiring initiation: the rewiring peer initiates a walker, which is actually a message carrying the necessary information about the rewiring peer.

Peer collection: the walker walks along peer connections, collects the information of the peers it accesses, and returns to the rewiring peer when its time to live (TTL) equals to 0.

Link update: the original peer selectively sets new links and discards old ones, according to the information of the explored peers.

Each peer repeatedly discovers more similar peers from its neighborhood (a set of peers that can be accessed within a given number of hops). The walker takes its steps according to a certain strategy and collects the information of the peers it explores. These peers are then used to update the peer's short-range links until the rewiring peer links to either semantically nearest peers (s-NN selection) [Voulgaris 2007] or the peers whose average similarity to the rewiring peer is below a given threshold (range-based selection) [Schmitz 2004, Raftopoulou 2008a].

Based on the formalization, peer rewiring is modeled as a decentralized local search approach to a combinatorial optimization problem, which refers to the task of building SONs. A Simulated Annealing (SA)-based decentralized local search approach is then employed. It guides the walker to take its steps to other peers according to a certain probability. The probability is controlled by two factors. One is the similarity between these peers and the rewiring peer; the other is a gradually decreasing parameter called temperature. When the temperature is high, the probability to step on any peer is almost the same, so peers are almost randomly explored in the neighborhood. As the temperature decreases, the probability to step on similar peers increases, and the probability to step on dissimilar peers decreases. Therefore, the strategy the walker employs gradually changes from random walk to greedy walk, which matches with the evolution of the network topology: from a random network to a SON. In other words, random walk is employed when the peers

are randomly connected; as peer clusters gradually emerge in the network, random walk is gradually replaced by greedy walk.

To rewire the links introduced or destroyed by dynamic behaviors, the walker is also guided by SA with an initial temperature decided by the context. If the quality of the short-range links does not improve because the current temperature is too low, the temperature is reset to a higher value to allow an extensive exploration in the neighborhood.

1.4 Contributions

We have three main contributions, which are described in the following.

1.4.1 Generic Models for Building SONs

We identify a generic framework for building SONs in self-organized P2P networks, where no central controller nor coordinators exist and frequent dynamic behaviors happen. We identify a generic peer rewiring pattern and formalize it as a three-step procedure that is initiated independently and periodically by each peer. Our technique provides a consistent framework for peer rewiring, while allows enough flexibility for the users/designers to specify its properties.

We model the building of SONs as a combinatorial optimization problem, and the process of rewiring peer connections as its decentralized local search solution. In the combinatorial optimization problem, an objective function is defined to measure the fitness of the SON, which involves the similarity between peers and their neighbors, and the number of accessible similar peers in their neighborhood. In the decentralized local search solution, each peer independently optimizes the configuration of its neighbors, by searching better configurations from a local search space (its neighborhood).

This optimization model reveals an explicit gap between building SONs (the combinatorial combination problem) and its state of the art solution: peer rewiring (the decentralized local search solution). Optimizing the configuration of peer's neighbors may not guarantee a global optimum of the combinatorial optimization problem: it depends on the specific local search strategy. This model is useful for better analyzing the state of the art approaches and designing a better decentralized local search strategy to reach the global optimum.

1.4.2 SA-based Decentralized Local Search Solution

We propose a novel Simulated Annealing (SA)-based decentralized local search solution to the combinatorial optimization problem described above. The approach

employs SA to implement an evolving local search strategy that matches with the evolution of the P2P overlay network topology. It extends the traditional way to use random search and greedy search individually or to integrate them with fixed probability [Schmitz 2004, Voulgaris 2007, Parreira 2007, Raftopoulou 2008a].

With the evolution of the P2P overlay network topology from a random connected network to a SON, peers' neighborhood structures are changing over the time. Consequently, it is more rational to allow more random searches in the beginning and more greedy searches in the end, since similar peers are gradually linked up in the neighborhood. The proposed SA-based approach implements this idea and takes the existing research one step further.

1.4.3 Extensive Experimental Study

We make an extensive experimental study about the effect of the network configuration and the local search configuration on the performance of building SONs and the subsequent IR. Specifically, we simulate different walking strategies, such as greedy walk, random walk, SA-based walk to figure out how they affect the performance of building SONs. The same strategies are used to discover similar peers for the new peers who join the network at different times (e.g., in the beginning where the network is still randomly connected; when similar peers start to become clustered; when the peers are well clustered). Their performance is studied and analyzed to find the relation between the strategy's performance and the joining time.

Moreover, we study the minimum number of links each peer should keep in order to maintain the whole network connected. A connected network can allow a peer to access all the other peers by following certain connections. It is the pre-requisite for peer rewiring and the subsequent IR task. With this minimum limit, we study how the number of links each peer keeps can affect the performance of peer rewiring and the subsequent IR. These experimental studies provide new insights into the fundamental issues related to network design and SON organization.

1.5 Organization of the Thesis

The thesis has six chapters. In Chapter 2, we give an overview of the state of the art about P2P Information Retrieval, SONs and the evaluation of the SON construction. Based on this overview, we point out the position of this thesis. In Chapter 3, we present the generic mechanism to build SONs and perform IR in SONs. Our optimization model of building SONs is then presented. Chapter 4 presents a SA-based decentralized local search solution for the combinatorial optimization problem we model in Chapter 3. Experimental results and analysis are presented in Chapter 5,

followed by conclusions and future works in Chapter 6.

Chapter 2

State of the Art

Contents

2.1 Introduction 9
2.2 Clarifying the Concepts 10
2.3 P2P Overlay Networks 12
2.4 P2P Information Retrieval 16
2.4.1 Global Indexing 17
2.4.2 Local Indexing 18
2.5 Semantic Overlay Networks (SONs) 19
2.5.1 SONs in Unstructured P2P Networks 20
2.5.2 Evaluation of SONs 25
2.6 Summary 27

2.1 Introduction

P2P networks are a type of overlay networks designed as an alternative to the conventional Client-Server infrastructure. In the latter infrastructure, servers are designed to offer services like storage and search to clients. The clients communicate with the servers to get services, and they do not offer any service. In P2P networks, instead, a peer acts both as a server and a client. When it offers services to other peers, the peer acts as a server; when it requests services from other peers, it acts as a client [Wang 2003].

Information Retrieval over P2P networks (P2P-IR) involves forwarding a query from its initiator to peers with relevant documents (query routing), performing local IR in these peers, and then returning the relevant documents to the peer that initiates the query. A lot of works in this field focus on how to achieve efficient query routing. To this end, the underlying overlay network matters a lot. Based on the same overlay network, the way to organize the resources matters too. This thesis specially focuses on IR in unstructured P2P networks, an overlay network where peers only know their neighbors and their dynamic behaviors do not cause much

workload on the network. In the lack of a central server, peers autonomously rewire their connections to form a SON that facilitates efficient P2P-IR performance (it will be described in Chapter 3).

In this chapter, we firstly introduce a list of concepts which will be repeatedly used in the thesis, then we review the state of the art in P2P-IR and SONs. We start from the introduction and analysis of different P2P overlay networks. We then give an overview of IR in P2P networks in which SONs play an important part. A detailed overview is presented about building SONs, specially building SONs in unstructured P2P overlay networks. In this detailed review, we also present how the documents of a peer are described and how similarity between peers is measured in the state of the art. Finally, we present metrics to evaluate the protocols to build SONs and the quality of the resulting SONs.

2.2 Clarifying the Concepts

Some concepts are clarified as follows. They will be repeatedly used in this thesis.

Overlay Network An overlay network is a computer network built on the top of another network, particularly a physical network [Jannotti 2000]. Nodes in the overlay are connected by virtual or logical links, each corresponding to a path–through one or multiple physical links–in the underlying network. **P2P overlay networks** are one of the typical examples.

P2P Overlay Network A P2P overlay network [Lua 2005] can be represented as a graph $G = \langle P, L \rangle$, where P is a non-empty countable set, called **peers**, and L is a set of pairs of different peers, called **links**. Throughout this thesis, we will refer to a peer as p_i by its order i in the set P. The link $l_{i,j}$ joins the peers p_i and p_j, which are defined as **connected**. The link is directed, and p_j is called the **neighbor** of p_i. For the sake of being concise, P2P networks will be used frequently to refer to P2P Overlay Networks.

Peers Peers refer to the nodes in a P2P overlay network. They have three roles in the network: (i) as resource provider to provide hardware resources (like processing power, storage capacity and network link capacity), services and content (text documents in this thesis); (ii) as resource consumers to access the resources provided by all the peers in the network; (iii) as message transmitters to forward a message to the next hop, to enable the communication between peers [Lua 2005]. Peers are also featured by their **autonomously dynamic behaviors**: (i) joining/leaving the network; (ii) changing their content without the permission of a central control point.

P2P Information Retrieval (P2P-IR) P2P-IR involves four steps: (i) query initiation: a peer issues a query when its user has an information request. We call this peer **query initiator**; (ii) query routing: a peer forwards the query to one or more of its neighbors once it receives a query, until the peers with relevant documents are reached or the time to live (TTL) of the query is equal to 0; (iii) local IR: when a peer receives a query, it performs IR against its text documents and sends the relevant documents back to the query initiator; (iv) postprocessing: once the query initiator receives these documents, postprocessing like ranking and replication removal can be performed to refine the results [Tigelaar 2012].

Semantic Overlay Networks (SONs) SONs are defined as a type of P2P overlay networks for organizing peers in thematic clusters with similar contents, so that queries can be selectively forwarded to only those peers having content within specific topics [Crespo 2002b]. In its original proposal, a global classification hierarchy was used to organize the thematic peer clusters, each corresponding to one class in the classification hierarchy and called a SON. However, SONs do not necessarily imply the use of semantics in the traditional sense (e.g., ontology). Peers can form clusters by connecting to the other similar peers [Tang 2003a, Voulgaris 2007, Raftopoulou 2009b]. The peer cluster generated in this way is not well defined as in the original proposal. Instead, all the peers are somehow connected with each other. For clarifying the presentation, in this thesis, we call the whole network with peers clustered a SON.

Peer Rewiring Peer rewiring refers to the process in which a peer removes an existing or dead link and builds a new one to another peer [Raftopoulou 2010]. A **rewiring peer** refers to the peer that performs the operation of removing/building links.

Neighborhood The neighborhood of a peer refers to the set of peers that can be accessed by this peer within a given number of hops. Specifically, if a peer p_i initiates a message with time to live (TTL) γ, and forwards it to other peers along all the possible links until its TTL equals 0, the peers the message visits are called the **neighborhood** of p_i. γ is called the **radius** of the neighborhood. The message that is forwarded along the possible links is also mentioned as a **walker** that walks along the possible pathes; forwarding the message to another peer is regarded as the walker going a **step** further.

Short-range Links are links that connect the peers within a thematic clus-

ter [Raftopoulou 2008a]. The number of short-range links is limited due to the cost for maintaining them. The peers the short-range links point to are called **short-range contacts**.

Long-range Links are links that connect peers of different clusters [Raftopoulou 2008a]. The number of long-range links is limited due to the cost for maintaining them. The peers the long-range links point to are called **long-range contacts**.

2.3 P2P Overlay Networks

In a P2P network, peers form a self-organizing network that is overlayed on the Internet Protocol (IP) network. Data is still exchanged over the underlying TCP/IP network, but in P2P overlay networks peers communicate with each other directly via the logical links (each of which corresponds to a path through the underlying physical network). In [Buford 2010], a list of properties are defined for a typical P2P overlay network. We list the most significant ones:

Resource sharing: each peer marks a part/all of its local resource as 'shared', as a contribution of the system resources.

Networked: peers are interconnected with other peers, so as to form a connected graph.

Decentralization: no central control point exists, and the behavior of the system is embodied by the collective behaviors of the participant peers. Some P2P systems however use a central server as a booster or a directory server of the system resources, e.g., the initial version of Napster[1] (it pioneered the idea of P2P file sharing with a centralized search facility).

Autonomy: behaviors of a peer in the P2P system are determined locally, and there is no single administration for the P2P system.

A typical P2P Overlay Network should also have the properties of *Symmetry*, *Self-organization* and *Scalability*. Symmetry implies that all the peers have equal roles; Self-organization lets peers to use local knowledge and local operations to collaboratively maintain the network architecture, no peer dominates the system; Scalability requires that the workload at each peer and the response time of the system do not grow more than linearly with respect to the overlay network size. In

[1] www.napster.com

addition, a P2P overlay network should be resilient against peers dynamic behaviors. For example, it should be able to facilitate new peers joining the network, and maintain the network stability when peers leave or change their resources. However, these properties are not exhibited in some P2P systems, according to the requirements of the specific application [Buford 2010].

With the above features, a P2P overlay network offers various services like routing architecture, search of data items, and fault tolerance [Lua 2005]. Based on how peers are connected to each other in the P2P overlay network, and how resources are indexed and searched, P2P overlay networks are classified as unstructured, structured, and hybrid.

Unstructured P2P Overlay Network: The notion of unstructured P2P overlay network describes a type of P2P overlay networks in which no global structure is imposed and peers randomly connect to each other. Peers locally index their resources and play equal roles. In order to search information, the query has to be forwarded from the initiator to the peers with relevant documents. Since peers has no global information about the network and the connections are random, the query has to be forwarded blindly or only using local information [Fletcher 2005]. A traditional way is flooding the query through the network, and blindly finding those peers that have the relevant information [Kalogeraki 2002].

Typical Examples of unstructured P2P overlay networks includes the initial versions of Gnutella[2] and FreeNet[3].

Structured P2P Overlay Network: Oppositely to unstructured P2P overlay network, a strict global structure is imposed on a structured P2P overlay network. Peers are arranged into a specific topology based on this global structure. Structured P2P overlay networks commonly implement a Distributed Hashing Table (DHT), in which a constant hashing function is used to assign each file to a particular peer. Thanks to the existence of the global structure, query forwarding can be performed in a deterministic way [Dhara 2010]. Chord [Stoica 2001] and CAN [Ratnasamy 2001] are two commonly used protocols in structured P2P network.

In Chord, a constant hashing function is used to generate an m-bit ID for each peer. All the peers form a Ring topology. The peers in the Ring are ordered by their IDs in a clockwise order. The same hashing function is used to generate an ID for each file. The ID is called *key*. Each peer keeps a hashing table for storing

[2] http://rfc-gnutella.sourceforge.net/
[3] https://freenetproject.org/

$\langle key, file\ information \rangle$ pairs. The *file information*, which contains the file's name and the IP address of the peer where the file is stored, is used to locate the file. A $\langle key, file\ information \rangle$ pair is stored in its first *successor peer*, defined as the peer in the Ring whose ID is equal to or follow the *key*. The *predecessor peer*, on the contrary, is the peer in the Ring whose ID is smaller than the *key*.

Each peer keeps a routing table to enable finding a file when its key is given. The routing table consists of the information of the other peers in the Ring, including the first predecessor of the peer, a list of its successors, and a finger table with m entries. The ith entry in the finger table points to the peer whose ID is the closest to $(id + 2^{i-1})\ mod\ 2^m$, with id as the ID of the peer. The searching procedure for a file location is as follows: upon receiving a lookup request (a key), the peer first checks if the key falls between its ID and its successor's ID. If it does, it returns the successor as the destination peer and terminates the searching service. If the key does not belong to the current peer, the peer forwards the request to the peer in its finger table with the ID the closest to and lower than the key. The forwarding process proceeds recursively until the destination peer is found.

In CAN, the IDs of peers and IDs (keys) of files are generated as a point in an m-dimensional space using a constant hashing function. Unlike in Chord where each peer stores the keys in an ID interval, peers in CAN store the keys in a region of the m-dimensional space. Specifically, the entire m-dimensional space is divided into zones where each node owns one zone. The node that owns a zone is responsible for the keys belonging to that zone. Similar to Chord, each peer keeps a routing table for lookup service. This table contains the information of its neighbors in the m-dimensional space. Neighboring nodes are the nodes whose zones are adjacent to each other. Given a key, searching the locations of the relevant files starts with forming a CAN message carrying the destination coordinates. The message is then forwarded toward the peer which owns the destination zone in a greedy fashion: the peer always forwards the message to the neighbor that is the closest to the destination.

Hybrid P2P Overlay Network: The unstructured and structured P2P overlay networks treat participant peers equally, and are referred as *pure/flat P2P overlay networks*. Hybrid P2P overlay networks, on the other hand, consider the heterogeneity of the participant peers. Powerful peers (e.g., with high storage, large bandwidth) are used as *super-peers*. They collaboratively perform the majority of the tasks, as a central server [Darlagiannis 2005]. So a hybrid overlay network is a combination of a flat P2P overlay network and a conventional Client-Server network.

Each super-peer manages a set of normal peers and holds a directory of their

resources. A normal peer can submits information request to the super-peer and get relevant documents returned. For the communication between super-peers, both unstructured and structured P2P infrastructure can be used [Eberspächer 2005]. Searching is performed in the following way. When a peer has an information request, it firstly sends the request to its super-peer. The super-peer then decides to forward this request to the other peers it manages or to the other super-peers managing other peers. If the relevant documents are located in the peers it manages, it takes the former action. It sends the request directly to the relevant peers. The peers receiving the request perform local IR and send relevant documents directly to the request initiator. Otherwise, the super-peer forwards the request message to the other super-peer(s) based on the routing mechanism of the super-peer overlay network. The other super-peers do the similar operation until the peers with relevant documents are found or the request time is out. An example of this type of hybrid P2P overlay networks is JXTA [Traversat 2003].

Analysis of P2P Overlay Network Architecture: Each of the above three overlay network architectures has its advantages and disadvantages. We analyze and compare them according to their tolerance to dynamic behaviors of peers, routing efficiency and the search service they can provide. The comparison summary is presented in Table 2.1.

Table 2.1: Comparison of different P2P overlay networks

	Tolerance to dynamic behaviors of peers	Search service	Routing efficiency
Unstructured	High	Specified by peers	Low
Structured	Low	matching(Chord); point/range query(CAN)	High
Hybrid	Normal peers: high Super-peers: low	Specified by peers	High

Unstructured P2P overlay network are highly robust against dynamic behaviors of peers, because peers are less dependent on each other as in structured P2P overlay network [Jin 2010]. Moreover, each peer can specify its local IR service, which allows more advanced searches for the coming queries. However, the lack of a global structure results in low routing efficiency, especially for unpopular resources [Buford 2010].

Structured P2P overlay networks have a higher routing performance comparing with unstructured P2P overlay network, due to its global structure and DHT. However, the structure imposed on the network requires a higher cost for topology maintenance, especially with frequent dynamic behaviors in the network. For example, in Chord, the joining or leaving action of a peer requires $O(\log^2 \textit{size of the network})$ messages [Stoica 2001]. In CAN, the same action requires $O(\textit{number of dimensions})$ messages [Ratnasamy 2001]. In addition, the global structure in structured P2P overlay networks limits the search service they can provide (only matching in Chord and point/range query in CAN).

While both unstructured and structured P2P overlay networks have their advantages and disadvantages in nature, hybrid P2P overlay networks combine most of their advantages, avoid some of their disadvantages and provide better performance [Schollmeier 2001], thanks to the employment of super-peers. Super-peers refer to the most powerful and stable peers in the network. They provide efficient routing performance. At the same time, the normal peers have more flexibility to join or leave the network without causing too much cost for maintaining the network topology. No global structure exists in the network, so advanced search service is possible, because the normal peers can specify the search services they provide. However, since hybrid P2P overlay network is composed of heterogenous peers, its application is limited to the situation where super-peers exist.

Although the different P2P overlay network architectures have different properties, they share a common operation called bootstrapping. It is executed when a peer joins the P2P overlay network for the first time. It is used to help the new peer to connect to the other peers in the network. A common approach for bootstrapping is through a bootstrapping server [Cramer 2004]. The bootstrapping server keeps a list of existing peers. When being contacted by a new peer, the bootstrapping server replies it with one or a set of randomly selected peers. The new peer contacts these peers and builds connections with them.

2.4 P2P Information Retrieval

To search documents stored distributively in peers, a query should be firstly issued by one peer. Then the query should be forwarded to peers where relevant documents are stored. The implementation of this task heavily depends on the P2P overlay network architecture. In Table 2.2, we classify P2P-IR into two categories according to their indexing techniques. One uses global indexing, and the other uses local indexing. Global indexing aims to use a consistent mechanism to index all the documents in the network. It can be implemented by a central server or by

employing structured P2P overlay network, for instance, Chord and CAN. While global indexing may require a well-organized P2P overlay network and perform IR efficiently, local indexing is implemented over unstructured P2P networks or super-peer P2P overlay networks. In local indexing, peers index their documents locally and perform local IR in order to find documents relevant to a query. The network topology is loosely organized because peers are not strictly dependent on each other as in structured P2P networks.

Table 2.2: Typical works for P2P-IR

Indexing approach	Overlay network	Typical works
Global Indexing	Chord	[Podnar 2007]
	CAN	[Tang 2003b]
Local Indexing	Unstructured P2P	[Crespo 2002b]
	Super-peer based P2P	[Doulkeridis 2010]

2.4.1 Global Indexing

The simplest way to implement global indexing is to use a central server. This approach avoids most problems regarding query routing and index placement. However, it has the problem of single point failure and other legal problems [Risson 2006]. The most famous example of this type of network is Napster. It is the pioneering P2P file sharing system whose original version ran into legal difficulties and then ceased operations.

Besides, Chord and CAN are two systems commonly used for global indexing in P2P overlay networks. In Chord based IR, documents are indexed by their component terms [Reynolds 2003]. The indices are stored in a HDT, where keys are the hashing function values of terms. To index a document in Chord, a set of encoded $\langle term, documentinformation \rangle$ pairs are published in a DHT. The document information refers to the name of the document as well as the IP address of the peer that kept it. When a query is issued, a hash value is generated for each of its component terms. These hashing values are used to query the DHT to get the the information of the documents that contain the terms. The relevant documents are those containing all the component terms of the query. Traffic cost is spent for publishing documents to DHT, querying the DHT, contacting the peers holding the relevant document, and transferring the documents to the query initiator. To save traffic cost for publishing documents, [Papapetrou 2007, Papapetrou 2010] form the peers into groups in a self-organized fashion. Instead of each individual peer submitting index information, all peers of a group cooperate to publish the

index updates to the DHT in batches. In [Podnar 2007], documents are indexed with highly discriminative keys, in order to save bandwidth consumption.

In CAN based IR [Tang 2003a, Tang 2003b], content-based full-text search is implemented via a distributed lookup table that supports multi-dimensional document/query representation. Documents in the network are indexed according to their vector representations (based on Latent Semantic Analysis: an advanced document indexing and ranking algorithm) in such a way that documents with similar vector representations are indexed in the lookup table of the same/adjacent peers. So the relevant documents of a given query can be found in the same/adjacent peers, which achieves both efficiency and accuracy.

2.4.2 Local Indexing

While global indexing has to be based on a global lookup table which requires a strictly organized network topology, local indexing can be implemented in a loosely organized P2P network. A peer simply indexes its local files and waits for queries from the other peers. When it receives a query, local IR is performed, and then the relevant documents are returned to the query initiator. Local indexing enables rich queries (not limited to a keyword lookup) and advanced retrieval technique.

To forward a query to target peers, a simple way is flooding. Normally, a TTL is defined to limit the hops the message is flooded, in order to save the traffic cost. However, it still generates a large volume of query traffic with no guarantee that a match will be found, even if it does exist in the network. There have been a lot of attempts to improve this flooding approach. For example: forwarding queries by a random walk [Lv 2002], an informed walk [Adamic 2003], or by clustering peers according to their content [Crespo 2002b] or interest [Sripanidkulchai 2003].

A variant, or rather an optimized methodology, of local indexing is aggregated local indexing. A super-peer based P2P overlay network is employed for this. A super-peer holds the index of both its own content as well as an aggregation/summary of the indices of all the peers it manages. This architecture introduces a hierarchy among peers and by doing so takes advantage of their inherent heterogeneity. It has been used by FastTrack and in recent versions of Gnutella. Query routing is performed by the super-peers which are in charge of sending the query to target peers. Since these super-peers have more computing power and high stability, communication among them has lower susceptibility to bottlenecks and thus query routing can be better performed compared to local indexing. For example, in [Balke 2005], a super-peer backbone is organized in the HyperCuP topology for optimizing the necessary query routing. In [Marin 2009], a super-peer based P2P network architecture is also used to improve the querying performance.

P2P-IR with local indexing avoids index publishing or updating, while that with global indexing must perform it. Thus, the cost for local indexing is cheaper, and the impact of churn on the network is lower. But for query routing, local indexing costs more than global indexing does, and makes unpopular data unreachable [Tigelaar 2012]. On the other hand, global indexing provides efficient query routing, but it only supports exact matching or point/range queries.

2.5 Semantic Overlay Networks (SONs)

SONs were firstly proposed and defined as overlay networks where nodes connect to other nodes that have semantically similar content [Crespo 2002b, Garcia-Molina 2003]. In SONs, nodes with semantically similar content are clustered together. This allows a flexible network organization that improves query performance while maintaining a high degree of node autonomy [Crespo 2002b]. The SONs in this proposal are organized based on unstructured P2P network architecture, which guarantees network flexibility and node autonomy.

The concept of SON is also used in P2P-IR proposals based on CAN and super-peer based architecture. For example, in [Tang 2003b, Tang 2003a], the authors propose an approach to cluster the documents using LSA and CAN-based P2P overlay network, so that semantically similar documents are indexed in neighboring peers. In [Doulkeridis 2008, Löser 2004, Kurve 2013], a super-peer based architecture is used to cluster the peers with similar content. Super-peers are used to manage the peer clusters. These works follow the principle of SONs to make peers with similar content neighbor to each other. They improve query performance, but allow a low degree of node autonomy due to either the global indexing as in [Tang 2003a, Tang 2003b] or aggregated local indexing as in [Doulkeridis 2008]. In other words, the documents in each peer are indexed and managed not only by the peer itself but also by the other peers.

Many studies show that peer search systems perform better when the content in peers are clustered and queries are sent to relevant clusters. In SON systems, a query are processed by firstly identifying the relevant peer clusters; then it is directly forwarded to those relevant clusters. Local search is only performed in the peers in those relevant clusters, so the search services of other peers can be freed for answering the other queries [Crespo 2002b]. Moreover, the peers in the same cluster can answer each other's query, if the query is about the same theme of the cluster [Sripanidkulchai 2003].

Based on the state of the art, we generalize a classification about P2P overlay networks and P2P-IR. Figure 2.1 illustrates this classification. It also illustrates the

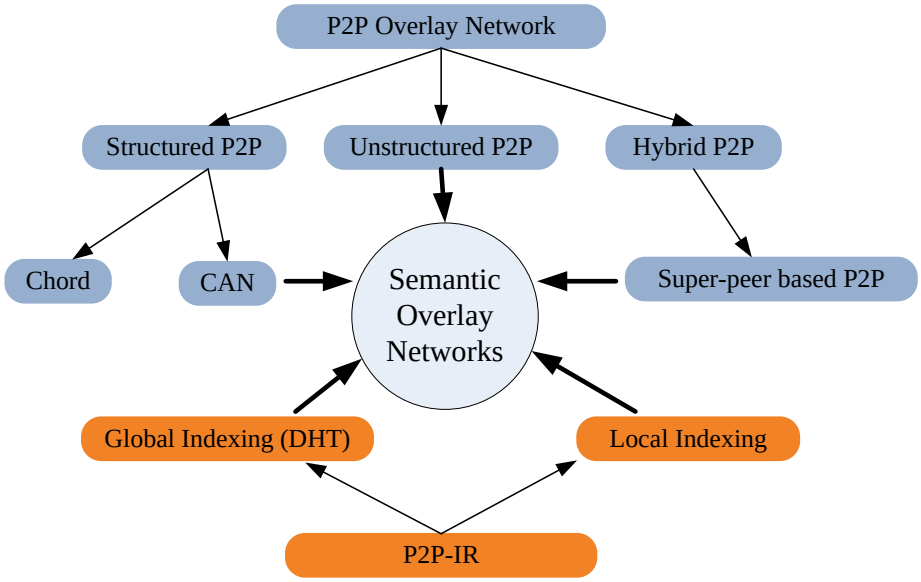


Figure 2.1: Categories of P2P overlay networks and P2P-IR, and the P2P network architectures and P2P-IR techniques employed by Semantic Overlay Networks (the thick arrowed lines)

overlay networks and P2P-IR approaches that are used for SONs. In this thesis, we focus on SONs based on unstructured P2P networks. This task is challenging due to its constraints: (i) peers have high autonomy over their documents; (ii) neither central server nor super-peers exist in the network; (iii) peers keep limited number of connections, so each peer only has a local knowledge about the network; (iv) dynamic behaviors should be tolerated, so that a robust protocol is required to maintain the topology of the SON. The first two constraints make it impossible to use any efficient global manipulation like structured P2P network/hybrid network infrastructure to organize the network into a SON. Instead, peers have to self-organize into clusters based on their local knowledge and local operations, while the local knowledge is constrained by the limited number of the connections each peer keeps.

The state of the art of SONs in unstructured P2P networks is presented in the following.

2.5.1 SONs in Unstructured P2P Networks

Three different techniques are commonly used to generate SONs in unstructured P2P networks: (1) one or more coordinators are used to cluster the documents in peers; (2) peers discovers their acquaintances with similar interest/content based on the feedbacks of the previous queries; (3) peers proactively discover the other

peers with similar interest/content. An comparison of these three techniques is summarized in Table 2.3.

Table 2.3: Overview of the techniques for generating and maintaining SONs

Coordinators	Query feedbacks	Proactive acquaintance discovering
Pros: peers can be well clustered; query routing can be efficiently performed. **Cons**: stable and powerful peers are required as coordinators.	**Pros**: no additional cost is required. **Cons**: accumulating querying feedbacks takes time; the pre-requisite assumption may be not realistic.	**Pros**: proactive similarity measuring based on unstructured P2P network **Cons**: high cost and bad cluster quality if the approach for acquaintance discovering is not appropriate.

2.5.1.1 Using Coordinators

In this technique, a central server or powerful and stable peers are required to be the coordinators. This technique can build SONs and perform the target tasks in SONs efficiently due to the global information as well as the facility the coordinators have, but its application is limited because heterogeneous peers are required. For example, a central controller is used in [Bawa 2003] to cluster the peers into topic-based groups each called segment. Two types of links between peers are defined, one for connecting peers whose documents are in the same segment, the other for connecting peers whose documents are in different segments. The central controller is a distinguished peer in charge of clustering documents and assign segment ID for new peers. Following a similar line, [Klampanos 2004] proposes a two-stage clustering procedure: individual peer documents are locally clustered using a hierarchical clustering algorithm; then a global clustering algorithm is performed in the controller based on the local clusters.

Coordinators are also used in [Doulkeridis 2006, Doulkeridis 2007, Doulkeridis 2008] where a hierarchy clustering approach is proposed based on peer coordinators to generate SONs. As in [Klampanos 2004], peers locally cluster their documents. Then the peers in local regions are grouped (called zones), forming clusters based on data stored on these peers. These zones are merged and clustered recursively until global zones and clusters are obtained. Local grouping and zone merging are performed by peer coordinators. In [Triantafillou 2003], a P2P architecture where nodes are logically organized into a fixed number of clusters

is presented. The information of clusters are calculated during the bootstrapping of the system and each node has the information of the clusters in the whole network.

2.5.1.2 Using Query Feedbacks

For this second technique, building SONs does not impose much workload on the system, because it takes advantage of the querying histories. A heuristic is used to detect the peers that share interests: peers that have content that we are looking for share similar interests [Sripanidkulchai 2003]. When a peer joins the system, it may not have any information about other peers' interests. Its first attempt to locate content is executed through flooding. The lookup returns a set of peers that store the content, which are the potential candidates of 'shortcuts'. As more lookups are performed, peers can build a list of 'shortcuts' that can be used to forward later queries. In [Sripanidkulchai 2003], a content location solution is proposed in which peers loosely organize themselves into an interest-based structure on top of the existing Gnutella network. A similar proposal is made in [Tempich 2004]. It defines a method for query routing that lets peers observe which queries were successfully answered by other peers. The peers memorize their observations, and subsequently use the information to select peers to forward requests to. [Sedmidubsky 2008] proposes two algorithms to improve the connection built based on the querying feedbacks as well as the later querying routing. One is used to manage query histories of individual peers with the possibility to tune the trade-off between the extent of the history and the level of the query-answer approximation; the other is to limits the exploration of the network in query routing.

In addition to find peers with similar interests, query histories can also be used to associate a peer with relevant peer clusters. [Koloniari 2008] models the cluster-reformulation problem as a game where peers determine their cluster membership based on potential gain in the recall of their queries. This work assumes that each cluster has a unique identifier, and that all the peers in the cluster are aware of this unique identifier.

However, accumulating query feedbacks takes time. When no queries or only a small number of queries are generated, IR has to be performed by flooding or random walk. Moreover, a peer may issue the queries that are not related with its content, hence the peers answering the queries can not be considered to share the similar interests to this peer. In this case, the query has to be forwarded in a blind way such as flooding or random walk. This problem can be observed in [Cholvi 2004]. The basic premise of this work is that file requests have a high probability of being fulfilled within the community they originate from, therefore increasing the search efficiency. To this end, peers perform local dynamic topology adaptations, based

on the query traffic patterns, in order to spontaneously create communities of peers that share similar interests. Similar to the other works mentioned above, dynamic topology adaptation is implemented by directing acquaintance links toward the peers that have returned relevant results in the past. A similar work is [Cohen 2007], where peers self-organize into overlapped groups by taking into account previous query satisfaction. Each peer can belong to different groups, by connecting to a set of peers that also belong to the same group. Hence, each groups can be regarded as an overlay, which is unstructured P2P network with peers containing similar items. Users decide which group to use for searching.

2.5.1.3 Proactively Discovering Acquaintances

Oppositely to the technique using querying feedbacks to build interest-based connection between peers, [Voulgaris 2007] proposes a proactive method to build semantic overlays. It is a two-layered approach combining two epidemic protocols, one allowing each peer to proactively find and dynamically maintain a list of similar peers; the other allowing each peer to maintain a list of random peers. For this technique, workload is required for proactively discovering peers with similar contents or similar interests, called *acquaintances*. Proactive acquaintance discovering is performed in an unstructured P2P network, so it allows peer autonomy, and tolerates dynamic behaviors.

However, if the way to discover acquaintances is not appropriately designed, more workload is needed, and the cluster quality is not be guaranteed as in the technique using coordinators. This factor is studied in [Raftopoulou 2008c, Raftopoulou 2008a]. The authors propose iCluster, which designs a mechanism to discover similar peers in one's neighborhood. The discovering is performed by forwarding a message along peer's connection. The peers the message visits are discovered peers. It can be implemented by forwarding the message randomly or to the most similar peer among the current accessible peers. Other possibilities to discovering similar peers are also proposed. Their performances are demonstrated according to simulated experiments, which shows that random walk can achieve the best cluster quality comparing to the other discovering approaches.

Another typical approach to proactively build SONs is proposed in [Parreira 2007]. It follows the spirit of peer autonomy and creates semantic overlay networks based on the notion of 'peer-to-peer dating'. Peers periodically select a peer to 'date' with, and decide weather or not to regard it as a 'friend' based on certain similarity estimation. A list of peers are managed as the 'dating' candidates, which include random peers provided by the underlying network infrastructure, the peer's current 'friends' as well as their 'friends'. These candidates are

probabilistically selected with respect to their types (random peers, 'friends' or the 'friends' of a 'friend').

The works in proactive acquaintance discovering differ not only in the specific protocols to discover the acquaintance, but also in the way they measure the similarity between two peers. A statistical language model is employed in [Linari 2006] to calculate the similarity. Peers in the network are represented by a statistical Language Model derived from their local data collections. A symmetrized and 'metricized' related measure, the square root of the Jensen-Shannon divergence, is used to approximate peer similarity before it is contacted. It hence maps the problem of forming SONs to a metric search problem. The peers periodically meet peers that are randomly picked or suggested by other peers. The search strategy exploits the triangular inequality to efficiently prune the search space and relies on a priority queue to visit the most promising peers first. In [Li 2008] where a framework called Semantic Small World (SSW) is proposed, peers and their documents are projected into a high-dimensional space generated by Latent Semantic Analysis (LSA). The similarity between two peers is then calculated based on their locations in this space. Thanks to LSA, this measurement can reflect the semantic similarity. However, the workload to use LSA is heavy because the LSA model has to be computed in a central point with the document collection of the whole network (or document samples) and updated as the content in the network changes. Besides, [Penzo 2008] and [Schmitz 2004] employ ontology to describe the content in each peer, and measure peer similarity based on their representative concepts. [Penzo 2008] addresses the issue of SON creation in a Peer Data Management System (PDMS) where peers have different schemas and they are connected through schema mappings. Each peer is represented by a set of concepts. Each concept will be associated to at most one SON. Heterogeneity is solved using the WordNet as background thesaurus. A distance function is defined among sets of concepts according to their relation in the ontology. A similar approach is also described in [Schmitz 2004].

Besides evaluating peer similarity based on their interests or contents, some works also consider other criteria. For example, [Löser 2007] proposes a system called INGA, where peers create and maintain shortcuts to other peers based on four layers: (i) at the content provider layer, shortcuts are created to remote peers which have successfully answered a query, (ii) at the recommender layer, information is maintained about remote peers who have issued a query, (iii) at the bootstrapping layer, shortcuts to well connected remote peers are kept, and (iv) at the network layer, connections are maintained to peers of the underlying network topology; [Parreira 2007] describes three measurements that can be used to identify good friends: (i) credits about peer's cooperation history, (ii) amount of the overlapped

documents between peers, and (iii) the semantic similarity of the contents of peers; [Bertier 2010] presents Gossple, an internet-scale protocol that discovers connections between users and leverages them to enhance navigation within the Web 2.0. A set of social anonymous acquaintances are discovered for each peer by a gossip protocol. The acquaintances are defined as the users that share the similar tags over their contents.

2.5.2 Evaluation of SONs

Semantic overlay networks have been studied mainly for subsequent tasks of information retrieval or information filtering, hence their evaluation has been oriented towards the performance of these tasks. In summary, the following aspects are evaluated: convergence speed to generate the SON, the quality of peer clusters, querying performance in SONs, and the workload required for generating the SON and querying.

In Figure 2.2, we present an overview about the evaluation metrics in the state of the art involving all the aspects. However, works that evaluate all the aspects are rare: usually the focus is only on some of the aspects. For example, only clustering quality and network workload are evaluated in [Eisenhardt 2003], since it aims to clustering documents rather than to P2P-IR. The evaluation of this work is performed by comparing the resulting document clusters with the ground truth. It also evaluated the time to take for one iteration of k-means clustering. Only workload is considered in [Banaei-Kashani 2004, Bawa 2003]. The former evaluates the proposed Small-World Access Methods (SWAM) by communication cost, computation cost, and query time. In the latter the quality of the proposed approach is evaluated by the average number of sites that are probed to answer a query and by the bandwidth and latency consumed for a query. Only IR performance is evaluated in [Doulkeridis 2007, Doulkeridis 2008], where the metrics of recall and precision are employed.

Most of the works focus on the evaluation of querying performance and workload in the network. Querying performance aims to evaluate the quality of the search results to a given query. The most commonly used metrics are recall and precision. Recall is defined as the fraction of relevant instances that are retrieved; precision is defined as the fraction of retrieved instances that are relevant. They are used in [Klampanos 2004, Bertier 2010]. In [Tempich 2004, Löser 2007, Sedmidubsky 2008], however, only recall is used as the metric for querying performance. Other works like [Li 2008, Sripanidkulchai 2003, Cholvi 2004] use search failure ratio and success rate. They use these metrics instead of recall and precision, because the searching task is designed to search the exact file given its name as a

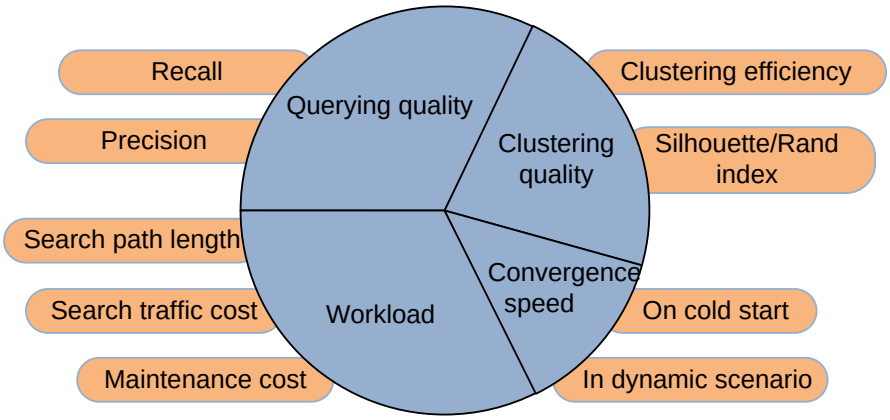


Figure 2.2: Overview of SONs evaluation, the size of each section implies the amount of corresponding works

query. Therefore, the search result is either successful or unsuccessful.

Workload, on the other hand, focuses on the traffic spent for building a SON and perform querying. To this end, various measurements are used in the literature. In summary, the following metrics are commonly used: (i) the average or minimum search path length to reply a query [Li 2008, Sripanidkulchai 2003, Cholvi 2004]; (ii) the number of messages required for answering a query [Li 2008, Sripanidkulchai 2003, Tempich 2004, Löser 2007, Raftopoulou 2008c]; (iii) maintenance cost for keeping the network robust against peers' dynamic behaviors [Li 2008, Bertier 2010]. Besides, in [Li 2008] index load is also evaluated, because the network topology in this approach is constructed by indexing similar documents in the same/adjacent peers. Peers adaptively self-organize themselves in order to maintain the topology; in [Cholvi 2004], a measurement called in-degree is employed to evaluate the number of the peers that chose one peer as their acquaintance.

Convergence speed is used to measure the time required to build a SON. In [Voulgaris 2007], the author used real world traces from the eDonkey file sharing system. The convergence speed on cold start, on adaptivity to changes in user interests is studied to evaluate the proposal. [Bertier 2010] evaluates the quality of a node's semantic neighbors through its ability to provide the node with interesting items. The author also considers the time required to build a network of Gossple from a random network, then considers the maintenance of this network by evaluating convergence in a dynamic scenario where nodes join an existing stable network.

Some works directly focus on the clustering quality—how well the similar peers

are clustered—rather than evaluate it according to the IR performance. For example, [Penzo 2008] evaluates the resulting clusters by their internal quality and external quality, which are quantified by Silhouette indices and Rand index respectively. Similarly in [Raftopoulou 2009a], several metrics are used to this end: clustering coefficient, generalized clustering coefficient, and clustering efficiency. The former two aim to quantify how the neighbors of a peer are connected with each other; the latter is rather designed for the subsequent task like IR; it measures how many similar peers a peer can access within certain number of hops.

The evaluations are often performed via two sets of experiments. One starts from a random peer configuration and examines whether the peer relocation protocol leads to the desired cluster configuration for the given data content and workload. The other starts from a 'good' cluster configuration for a given content and workload, and examines how well the periodic reformulation protocol adapts to changes of the content and the workload [Koloniari 2008, Bertier 2010, Voulgaris 2007, Raftopoulou 2010].

To summarize, the evaluation works in the state of the art mainly focus on the performance of IR in generated SONs, such as IR recall/precision and searching cost. Comparatively, the quality of peer clusters and the convergence speed to build SONs are less considered. However, if we consider SONs as a general P2P network topology that can be used not only for IR but also other applications like recommendation [Kim 2008], it is more meaningful to directly evaluate the quality of peer clusters to demonstrate its potential performance on the target task. The convergence speed also matters, because it indicates the time a peer has to wait until peer clusters emerge in the network and benefit the peers's target task like IR.

2.6 Summary

An overview and analysis of P2P-IR and SONs were presented in this chapter. We briefly introduced P2P overlay networks, and gave an analysis about these P2P overlay networks. We then introduced P2P-IR, focusing on how the contents in the network are organized and indexed. Among the works on P2P-IR, SONs outperform the others with respect to their performance in target tasks like IR.

We reviewed the methodology to build SONs in unstructured P2P network, which is commonly implemented by using powerful and stable peers as coordinators, feedbacks of querying history, or proactive acquaintance discovering. The application of the first approach is limited in the network with a certain number of powerful and stable peers. Given an unstructured P2P network where peers have equal roles, this approach is not suitable to reorganize it into a SON. The second approach

builds SONs in such a passive way that acquaintances may not be found quickly. Moreover, the premise of this approach—two peers share similar interests if one previously answered the other's query—is not always true in the reality, considering the fact that a peer may issue casual queries which are not relevant to its interests. In the third approach of proactive acquaintance discovering, no constraints are imposed on the peers and building SONs can be done efficiently if the right discovering strategy is used. However, works using this approach independently demonstrate their acquaintance discovering strategies. A generic framework is missed to formalize the principles shared among these works. These shared principles will be useful for studying the essential point of this approach. It would be also useful for designing a better discovering strategy, so that the decentralized discovering behaviors can more efficiently generate a network topology with globally better peer clusters.

An overview of the evaluation metrics for SONs was presented in the end. Many works focus on the evaluation of the IR performance in SONs and the workload of building the SONs. However, a direct evaluation of the cluster quality in a SON may be more worthwhile, considering that a good cluster quality would benefit not only IR, but also the other target tasks like collaborative filtering and recommendation.

Chapter 3

Optimization Model for Building SONs

Contents

3.1 Methodology 30
3.1.1 Similarity Measurement 30
3.1.2 Definition of Links 31
3.1.3 Building SONs via Peer Rewiring 33
3.1.4 Searching in SONs 37
3.2 Building SONs as an Optimization Model 39
3.2.1 Building SONs: Combinatorial Optimization Problem 39
3.2.2 Peer Rewiring: Decentralized Local Search Solution 41
3.2.3 Discussion: From Decentralized Local Search to Global Optimization 47
3.2.4 Related Work 48
3.3 Summary 49

In this chapter, we present a methodology to build SONs in unstructured P2P networks. Briefly, the connections between peers should be configured in such a way that similar peers are linked up and thus form a cluster. Since there is no centralized controller, the configuration is performed in such a way that peers autonomously rewire their links by removing the old links and building new ones to peers with more similar contents. To do this, a peer must discover the peers with more similar contents from its neighborhood, and uses them to replace its less similar neighbors. In Section 3.1, we identify and formalize a generic procedure of peer rewiring, and then describe the process of IR in the resulting SONs. In Section 3.2, we describe our optimization model to build SONs. The task of building SONs is modeled as a combinatorial optimization problem, and peer rewiring is modeled as a decentralized local search solution. Based on this model, we study how a decentralized local search approach (peer rewiring) can result in a global optimum of the system (SON).

3.1 Methodology

To cluster peers with similar content, a similarity metric must be defined. In this section, we firstly introduce the similarity measurement between peers, and then present the generic procedure of peer rewiring. Finally we introduce IR in the achieved SON.

3.1.1 Similarity Measurement

To evaluate the similarity between peers, each peer is represented by a formal description of its content. We call it the peer profile, in the similar way as [Bertier 2010]. Depending on the target application, a profile can be a term vector that summarizes the frequent terms in the peer's content [Raftopoulou 2008a], a set of tags the user employs to annotate the content [Bertier 2010], or topics/concepts derived from an ontology [Schmitz 2004] or machine learning techniques like Latent Dirichlet Allocation (LDA) [Draidi 2011]. A peer may have multiple profiles, each corresponding to one of its interests. Since the number of profiles has no significant effect on our study, we assume that each peer p_i only has one interest. So each peer only has one profile. The similarity between two peers is calculated as the semantic similarity between their profiles.

Comparing to a peer profile described with terms/tags, topics can exhibit the semantics in the content; comparing to the topics derived from an ontology, LDA [Blei 2003] is a more generic approach to model the topics in documents. So we follow the idea in [Draidi 2011] to represent the profile of a peer p_i as a set of topics. Formally, it is described as: $Prof_{p_i} = \{topic_1^i,\ topic_2^i,\ topic_3^i ...\}$, where each element refers to the ID of a topic. To implement this, LDA is used to model all the topics in the network. The model is in turn used to infer the topics associated with an individual peer. More details about this implementation are provided in Chapter 5.

We employ Jaccard distance [Levandowsky 1971], one of the classic measurements for the distance between two sets, to evaluate the similarity between peer profiles, as showed in Equation 3.1. A small Jaccard distance refers to high similarity between two peers. In the rest of the thesis, we will interchangeably use 'similarity' and 'distance' when referring to the metric of Jaccard distance.

$$Dis(p_i, p_j) = 1 - \frac{|Prof_{p_i} \cap Prof_{p_j}|}{|Prof_{p_i} \cup Prof_{p_j}|}. \tag{3.1}$$

Jaccard distance is a symmetric metric which satisfies the triangle inequality [Levandowsky 1971]: $Dis(p_a, p_b) + Dis(p_b, p_c) \geq Dis(p_a, p_c)$. In other words, if $Dis(p_a, p_b) + Dis(p_b, p_c) < x$, $Dis(p_a, p_c)$ is also below x. Given $x < 1.0$, these

three peers are similar to each other with a certain value of similarity. The smaller x is, the more similar they are. With this feature, peers with similar contents can be clustered. This will be useful in building SONs, as stated in [Linari 2006].

Besides the above techniques to generate peer profile and calculate peer similarity, other techniques can also be used, if they satisfy the following properties : (i) the representation of peer profile and the similarity measurement exhibit the semantics in the documents; (ii) the similarity measurement satisfies the triangle inequality. In this thesis, we use LDA and Jaccard distance to calculate peer profile and peer similarity, because they embody the properties described above. It is out of the scope of the thesis to study the different techniques.

3.1.2 Definition of Links

To build SONs in unstructured P2P networks, each peer p_i maintains a set of links to the peers that have similar contents to p_i. In [Linari 2006, Voulgaris 2007], the links point to k most similar peers; in [Raftopoulou 2008a, Schmitz 2004], they point to a set of peers whose average similarity to p_i is above a predefined threshold; in [Sripanidkulchai 2003], these links point to the peers that answered p_i's queries, since the authors assume that peers share the similar interests if one can answer the other's queries. Besides maintaining links to the similar peers, in [Raftopoulou 2008a], each peer also maintains a set of links pointing to the peers with different contents, in order to keep the whole network a connected component. For the same object, peers in [Voulgaris 2007] keep a set of links to some random peers. Similarly, in [Linari 2006, Sripanidkulchai 2003], an underlying random overlay network is used to keep the whole network connected.

Although building SONs mainly aims to cluster similar peers up, maintaining the whole network connected is also important. A connected network can offer an underlying infrastructure to discover similar peers in the network. After a SON is built, the whole network should still be connected, so that peer clusters can communicate with each other.

In this thesis, we follow the way used in [Raftopoulou 2008a] to define the links into *short-range links* and *long-range links*: the former point to the similar peers; the latter point to peers with different contents. These peers are called *short-range contacts* and *long-range contacts*, respectively. Formally, p_i's short-range contacts and long-range contacts are represented as P^i_{short} and P^i_{long}. Short-range links are used to cluster semantically similar peers, while long-range links are used to keep peer clusters connected and thus provide paths between peer clusters. Regarding IR, short-range links are used to search information within a peer cluster, while long-range links are used to forward the query to the relevant peer clusters.

Each peer p_i keeps the information of its contacts in a routing table, where each entry stores the information of one contact. The entry contains the type of the contact (Short-range or Long-range), its IP address, its peer profile, its similarity to p_i, the last time it was contacted, and other application-related information if necessary. An example of routing table is given in Table 3.1:

Table 3.1: Example of a routing table

	Type	IP	Profile	Similarity	Last contact
Link A	Short-range	IP_{p_a}	$Prof_{p_a}$	0.2	14:59,10/18/2013
Link B	Long-range	IP_{p_b}	$Prof_{p_b}$	0	14:00,10/18/2013

For each peer p_i, the number of its contacts in P^i_{short} and P^i_{long} is represented as s and l. The values of s and l are pre-fixed by the network designer or the user. A threshold θ is defined to distinguish the short-range contacts P^i_{short} and long-range contacts P^i_{long}. Formally, the short-range contacts of p_i should meet the following requirement:

$$P^i_{short} = \left\{ p_j \;\middle|\; \frac{1}{|P^i_{short}|} \sum\nolimits_{p_j \in P^i_{short}} Dis(p_i, p_j) \leq \theta \right\}, \tag{3.2}$$

where $\frac{1}{|P^i_{short}|} \sum_{p_j \in P^i_{short}} Dis(p_i, p_j)$ refers to the average distance between p_i and its short-range contacts. It is called *intra-cluster distance*. To facilitate the presentation, we formally define the intra-cluster distance between p_i and its short-range contacts as follows:

$$IntraDis(p_i) = \frac{1}{|P^i_{short}|} \sum\nolimits_{p_j \in P^i_{short}} Dis(p_i, p_j). \tag{3.3}$$

Based on the same threshold θ, long-range links should respect the following constraint:

$$P^i_{long} = \{p_j \,|\, Dis(p_i, p_j) > \theta \text{ and } Dis(p_j, p_k) > \theta,\, p_k \in P^i_{long} - \{p_j\}\}, \tag{3.4}$$

where the distance among p_i and the clusters it links to is above the threshold θ. This constraint makes the peer clusters with different content connected and avoids the redundancy in the long-range links, such that there do not exist two long-range links pointing to the same peer cluster.

The reason of using θ is to control the quality of peer clusters formed by short-range links. Its value must be decided carefully, because a very large θ results in loose clusters while a very small θ makes it difficult to connect similar peers

together. This can affect the subsequent query routing performance, as explained in [Schmitz 2004, Raftopoulou 2008a].

3.1.3 Building SONs via Peer Rewiring

When a peer joins an unstructured P2P network, it initializes its short-range links and long-range links by connecting to $s + l$ random peers that are provided by a bootstrapping server. For building SONs in unstructured P2P networks, peers must rewire their short-range links to similar peers. Long-range links can also be rewired along the same procedure. The protocols of peer rewiring have been proposed and framed diversely in the state of the art. A generic methodology is required to provide a consistent framework for this task, while allow enough flexibility for the protocol designer to specify its properties.

In this subsection, we identify and formalize peer rewiring as a periodical three-step procedure. A single three-step procedure is called a *rewiring cycle*, involving the steps of rewiring initiation, peer collection and link update. The peer that initiates the rewiring process is called *rewiring peer*. For each peer p_i, the object of its periodical rewiring cycles is to discover more similar peers in the neighborhood by a *walker*, to use them to replace current less similar short-range contacts, and finally to achieve such links that satisfy the requirement of short-range contacts.

3.1.3.1 Rewiring Initiation

In initiating step, the rewiring peer p_i firstly checks its current links. According to the situation of current links, it decides weather or not to initiate a walker, for exploring p_i's neighborhood and collecting the information of peers in the next step. If p_i's current links meet the predefined criteria and all of them are online, p_i does not initiate the walker and the rewiring cycle terminates.

The walker is actually a message R_i called *rewiring message* initialized by the rewiring peer p_i. It is described as a tuple $\langle IP_{p_i}, Prof_{p_i}, TTL_{R_i}, C\rangle$:

- IP_{p_i}: IP address of the rewiring peer p_i;
- $Prof_{p_i}$: profile of p_i;
- TTL_{R_i}: time to live (TTL) of the message R_i;
- C: an initially empty list for storing the information of collected peers.

The properties that can be specified by the designer include: the value of TTL_{R_i}, the other data structure included in the tuple to carry more information, the criteria of good links, and the exception management. The exception refers to the case where

the short-range links of some peers never meet the criteria. For example, a peer may never achieve a set of short-range contacts with intra-cluster distance below the threshold θ, because such peers do not exist in the network. In this case, the peer can either continue to perform peer rewiring or stop the process. The former choice is employed in this thesis. For the criteria of links, we use Equation 3.2 and 3.4 in this thesis.

3.1.3.2 Peer Collection

Peer collection aims to discover more similar peers in p_i's neighborhood and then to collect their information, in order to update p_i's current contacts in the next step. This is implemented by a walker that walks along the links and collects the information of the peers it accesses. The walking process is actually a procedure in which the rewiring message R_i is forwarded from one peer to another. We regard it as a walker walking along a path, for the sake of both clear presentation and later usage. During the walking process, information about the visited peers is recorded in $R_i.C$ (as well as the neighbors of the visited peers, depending on the specific rewiring protocol).

When a peer p_j receives a rewiring message R_i, it firstly reduces its TTL by 1. It then calculates $Dis(p_i, p_j)$ and appends a tuple $\langle IP_{p_j}, Prof_{p_j}, Dis(p_i, p_j)\rangle$ to the element C of R_i. p_j then forwards R_i to one of its neighbors (or n of its neighbors, depending on the specific rewiring protocol). The forwarding process continues recursively until TTL_{R_i} becomes 0. When TTL_{R_i} equals to 0, the rewiring message is sent back to p_i. Each time the rewiring message is forwarded from one peer to another peer, it can be considered as the walker takes its step to the next stop.

The key to peer rewiring is the strategy to choose the peers from a set of candidates (p_j's neighbors) and forwards R_i to them. In general, different strategies may differ from each other with respect to their performance in the efficiency and effectiveness of building SONs. This will be studied in Chapter 4.

3.1.3.3 Link Update

Link update is performed in p_i after it receives the returned rewiring message R_i. According to the information of the peers collected in $R_i.C$, p_i selectively sets new links and discards old ones.

To update short-range links, old links corresponding to less similar peers are replaced with new ones corresponding to more similar peers. The implementation details are showed in Algorithm 1. Firstly, the collected peers $P^i_{collected}$ as well as the short-range contacts P^i_{short} are ordered according to their distance to the rewiring

peer p_i (line 2–3). Then the peers with lowest value of distance in $P^i_{collected}$ is used to replace the short-range contacts with the highest value of distance, if only the former distance is lower than the latter (line 6–7). REPLACE($P^i_{short}[n]$, $P^i_{collected}[n]$) replaces the information of $P^i_{short}[n]$ in p_i's routing table with the information of $P^i_{collected}[n]$.

Algorithm 1: Updating short-range contacts of p_i

```
1  Input: P^i_collected, P^i_short; // collected peers and current short-range
       contacts, |P^i_short| = s
2  SORT(P^i_collected); // increasingly by their distance to p_i
3  SORT(P^i_short); // decreasingly by their distance to p_i
4  n = 0; // indicate the position of the peers that are being
       processed
5  while n < P^i_collected.length and n < P^i_short.length do
6      if Dis(P^i_collected[n], p_i) < Dis(P^i_short[n], p_i) then
           // update the entry of P^i_short[n] in p_i's routing table with
              P^i_collected[n]
7          REPLACE(P^i_short[n], P^i_collected[n]);
           // move to the next collected peer and next short-range
              contact
8          n++;
9      else
10         break;
11     end
12 end
```

Long-range links of p_i can also be updated with the information of the collected peers. The details to update long-range contacts are presented in Algorithm 2. The idea is to replace a current long-range contact with a collected peer, in such a way that the collected peer's distance to p_i and the other long-range contacts is above the threshold θ. Specifically, we pick up the collected peers one by one. For each collected peer p_m, we check its distance to p_i and p_i's long-range contacts (line 4–9), and record the long-range contact in *peerToUpdate* if the distance is below or equal to θ. p_m will not be used to update p_i's current long-range contact if more than one long-range contact is recorded in *peerToUpdate*, because replacing one of them with p_m still results in more than one link pointing to the same peer cluster, and this does not make any improvement to the configuration of the long-range contacts

defined in Equation 3.3. If *peerToUpdate* is empty, it implies that p_m is from a peer cluster which is different from all the peer clusters p_i's current long-range contacts belong. In this case, a random long-range contact is selected and replaced by p_m just for refreshing the links.

Algorithm 2: Updating long-range contacts of p_i

```
1  Input: P^i_collected, P^i_long; // collected peers and current long-range
       contacts, |P^i_long| = l
2  for p_m in P^i_collected do
3      peerToUpdate = {};
4      if Dis(p_m, p_i) > θ then
           // check p_m's distance to p_i's other long-range contacts
5          for p_n in P^i_long do
6              if Dis(p_m, p_n) ≤ θ then
7                  peerToUpdate = peerToUpdate ∪ {p_n};
8              end
9          end
           // p_m does not replace the current long-rang contact if
              |peerToUpdate| > 1
10         if |peerToUpdate| == 1 then
11             REPLACE(p_n, p_m); // p_n is the peer recorded in
                   peerToUpdate
12         end
           // randomly replace a long-range contact when peerToUpdate
              is empty
13         if |peerToUpdate| == 0 then
14             Randomly select a peer p_n from P^i_long;
15             REPLACE(p_n, p_m);
16         end
17     end
18 end
```

Long-range links can also be updated by initiating a special rewiring cycle. In this special rewiring cycle, the rewiring message is forwarded either randomly or to the peer whose distance to the rewiring peer is above θ. When the rewiring message is returned to p_i, the same algorithm 2 is used to update p_i's long-range links. The long-range links are initially random and then connect the peers with

different content, peers always can access peers with different contents. Moreover, there are usually much more peers with different contents than peers with similar contents in the network. These factors make it much less difficult to discover peers with different contents than finding similar peers, so we will not focus too much on rewiring long-range links in the rest of the thesis.

To maintain the links, each peer periodically sends a 'hello' message to its contacts, to check if the contacts are still online or changed their profiles. Note that in the above algorithms, contacts are updated by only considering their distances to the rewiring peer, as well as to each other when updating long-range contacts. Updating the contacts that are not online is not presented, because it is a trivial task. But in the real application, this should be considered.

The generated SON is evaluated with respect to clustering efficiency and its performance for target tasks like IR and information filtering. A good SON should allow a peer to access its similar peers within one or few hops along the short-range links. It should also provide good performance for the subsequent tasks.

3.1.4 Searching in SONs

Searching in SONs involves (i) forwarding a query from its initiator to relevant peer cluster(s), and (ii) finding relevant documents stored in the peers of the cluster(s). The first step is called *inter-cluster search* and the second step *intra-cluster search*.

The searching is performed via an approach similar to the line in [Raftopoulou 2008a]. Specifically, when a peer p_i initiates a query q_i, it also computes a topic-based representation for it. The query is integrated in a query message Q_i, which contains the following information:

- IP_{p_i}: IP address of the query initiator p_i;
- q_i: the query issued by p_i, which is used to perform local IR in relevant peers;
- $Prof_{q_i}$: topic-based representation of q_i;
- TTL_{Q_i}: time to live of this query message;
- D_{q_i}: an initially empty list for storing the information of the relevant documents.

When a query is initiated by a peer p_i, one or more relevant peer clusters are discovered firstly, as showed in Algorithm 3. Specifically, when a peer p_k receives a query (p_i is also considered as such a p_k), it calculates the distance between its profile and that of the query. If the distance is below or equal to θ, we consider

that p_k is the member of a relevant peer cluster. Through p_k, intra-cluster search is performed as described in Algorithm 4. Otherwise, p_k forwards the query to n_q of its contacts which are most similar to Q_i. The forwarding process continues until TTL_{Q_i} equals to 0 or p_k is the member of a relevant peer cluster.

Algorithm 3: INTER_CLUSTER_SEARCH(p_i,Q_i)

```
1  p_i initiates a query Q_i = ⟨IP_{p_i}, q_i, Prof_{q_i}, TTL_{Q_i}, D_{q_i}⟩;
2  while TTL_{Q_i} > 0 do
3      for each peer p_k that hosts Q_i do
           // Prof_{q_i} is used to calculate this distance
4          if Dis(p_k, q_i) ≤ θ then
5              INTRA_CLUSTER_SEARCH(p_i,Q_i);
6          else
7              TTL_{Q_i} = TTL_{Q_i} − 1;
8              p_k forwards Q_i to n_q contacts that are the most similar to q_i;
9          end
10     end
11 end
```

Algorithm 4: INTRA_CLUSTER_SEARCH(p_i,Q_i)

```
1  Q_i = ⟨IP_{p_i}, q_i, Prof_{q_i}, TTL_{Q_i}, D_{q_i}⟩;
2  Reset TTL_{Q_i} as k_{q_i};
3  while TTL_{Q_i} ≥ 0 do
4      for each peer p_k that receives Q_i do
5          p_k performs local IR and returns the relevant documents to p_i or save
           them in D_{q_i};
6          TTL_{Q_i} = TTL_{Q_i} − 1;
7          if TTL_{Q_i} < 0 and D_{q_i} is not empty then
8              return D_{q_i} to p_i
9          else
10             p_k forwards Q_i to its short-range contacts whose distance to q_i is
               below or equal to θ;
11         end
12     end
13 end
```

Note that when the distance between the query and the initiator p_i is below or equal to θ, we consider the initiator p_i is the member of a relevant peer cluster to the query. In this case, the query will be diffused along p_i's short-range links.

Once a relevant peer cluster is found at p_k, intra-cluster search is performed. p_j spreads the query to the peers that its short-range links connect to. The TTL_{Q_i} is reset to be a smaller value than the initial TTL_{Q_i} in Algorithm 3, because the peers in a cluster can be accessed within less hops. A simple and intuitive way to spread the query is flooding, as presented in [Raftopoulou 2008a]. However, flooding the query in a large peer cluster is expensive. Moreover, some peers may not have the relevant documents even though they are in the same cluster of p_k. Information like querying history can be introduced to improve the performance [Tempich 2004], but it takes time to accumulate query feedbacks. In this thesis, we employ a more intuitive approach, as described in Algorithm 4. The idea is to forward the query only to the peers whose distance to the query profile is below or equal to θ. When a peer in the cluster receives a query, it performs local IR and returns the relevant documents to the query initiator p_i. A document is relevant to the query if its similarity to the query is above a threshold. More details about this will be explained in Chapter 5.

3.2 Building SONs as an Optimization Model

3.2.1 Building SONs: Combinatorial Optimization Problem

In previous section, we have built SONs by connecting similar peers via short-range links and peers with different contents via long-range links. In this section, we model this process as a combinatorial optimization problem in a graph $G = \langle P, L \rangle$. G is the graph representation of the P2P network. P is the peers in the network, and L is the short-range links in the network. Given a number of peers and the number of short-range links for each peer, the task is to find the optimal graph configuration such that the similar peers are clustered up and then similar peers can access each other in one or few hops. Long-range links are not considered in this model, because (i) they do not imply the quality of peer clusters; (ii) it is not difficult to achieve optimal long-range links to keep the peer clusters and thus the whole network connected (refer to Section 3.1.3.3).

Formally, the optimal graph configuration should be the one satisfying the following two properties:

Property 1 Intra-cluster distance at each peer should be below or equal to a threshold θ, following Equation 3.2. This property aims to control the quality of the peer's short-range contacts.

Property 2 Each peer should be able to access all the other similar peers within a few hops. To simplify, similar peers with a distance below or equal to θ should be accessible within γ hops.

To quantify these properties of the graph, an objective function is defined with the following principles:

Principle 1 The value of the objective function should characterize the fitness of the graph configuration.

Principle 2 The objective function should be a proper integration of two parts, each corresponding to one of the two properties of the graph configuration.

We qualify the property 1 for each peer with the following equation:

$$\overline{IntraDis}(p_i) = \begin{cases} 0, \text{ if } IntraDis(p_i) \leq \theta \\ \theta - IntraDis(p_i), \text{ if } IntraDis(p_i) > \theta \end{cases} \tag{3.5}$$

where $\overline{IntraDis}(p_i)$ is called the *relative intra-cluster similarity* of p_i. Ideally, the optimum of this equation is 0. It corresponds to the fact that p_i has obtained the optimal short-range contacts: the intra-cluster distance at p_i is below or equal to threshold θ. In reality, the optimum of this equation may be negative for some peers, when there do not exist such peers in the network whose average distance to p_i is equal or below to θ. But this will not affect the property of this equation in measuring the fitness of p_i's short-range contacts.

For each peer, property 2 is qualified as:

$$\overline{C}(p_i) = \frac{|\{p_j \,|\, H(p_i, p_j) \leq \gamma, Dis(p_i, p_j) \leq \theta\}|}{|\{p_k \,|\, p_k \in G.P, Dis(p_i, p_k) \leq \theta\}|}. \tag{3.6}$$

where $H(p_i, p_j)$ refers to the number of hops from p_i to p_j. This is called *clustering efficiency*, proposed by [Raftopoulou 2008b] to quantify the connections of the similar peers. It is calculated as the number of peers $\{p_j\}$ that can be reached from p_i within γ hops following short-range links and whose distance to p_i is below or equal to θ, divided by the total number of peers in the network whose distance to p_i is below or equal to θ.

This metric is similar to the recall metric used in IR[1]. Another metric similar to the precision metric[2] can also be used, which is calculated by replacing $|\{p_k \,|\, p_k \in G.P, Dis(p_i, p_k) \leq \theta\}|$ with $|\{p_k \,|\, H(p_i, p_k) \leq \gamma\}|$ in Equation 3.6. Both of these two measurement focus on the similar peers that can be accessed within γ hops and

[1] Recall is defined as the fraction of relevant instances that are retrieved.
[2] Precision is defined as the fraction of retrieved instances that are relevant.

whose distance to p_i is below or equal to θ. But recall evaluates if p_i can access all the peers in the network whose distance to p_i is below or equal to θ, while precision evaluates if all the peers that p_i can access are similar to p_i (e.g., with a distance below or equal to θ).

We focus on the recall in this thesis. For each peer p_i, the recall can show if most of its similar peers are clustered in its neighborhood. This is important for the target task like IR, because a higher recall can allow the queries to be diffused, within few hops, to most of the peers that are similar the query and thus have the potential to answer the queries.

Relative intra-cluster similarity $\overline{IntraDis}(p_i)$ and clustering efficiency $\overline{C}(p_i)$ are used to qualify the property 1 and 2 for each peer. They are complementary to each other: the former measures the local cluster quality between a peer and a limited number of short-range contacts; the later shows how each peer connects to all the peers that are similar to it. By considering the P2P network as a single system with component peers, we define the objective function as:

$$O(G) = \frac{1}{N}\sum_{i=1}^{N}(\overline{IntraDis}(p_i)) + \frac{1}{N}\sum_{i=1}^{N}(\overline{C}(p_i)), \tag{3.7}$$

where N represents the total number of peers in the network. The two components of this object function are called relative intra-cluster similarity of the network and clustering efficiency of the network, respectively. Ideally, the maximum value of $\frac{1}{N}\sum_{i=1}^{N}(\overline{IntraDis}(p_i))$ should be 0, which means that intra-cluster distance of all peers is below or equal to θ. In reality, however, there may be peers such that their average distance to the most similar peers is larger than θ. This can happen when the peer profiles are not uniformly distributed in the topical space. In that case, the optimum of this function would be smaller than 0. The maximum value of $\frac{1}{N}\sum_{i=1}^{N}(\overline{C}(p_i))$ is 1. An optimal network configuration should be the one optimize both of the two components of the object function. However, a network with the optimal relative intra-cluster similarity does not necessarily imply that it also has an optimal clustering efficiency (will be discussed in Section 3.2.3), so we keep this two components to be independent by simply summing them in the objective function.

3.2.2 Peer Rewiring: Decentralized Local Search Solution

In order to optimize the objective function of the graph configuration in the previous subsection, an intuitive way is to exhaustively enumerate the possible graph configurations and calculate their objective function values. Thus the highest value will be found as the optimum of the objective function. The corresponding graph configuration is the optimum configuration.

However, exhaustive enumeration of all possible graph configurations is not possible, because the number of possible graph configurations is too large. For a network with N peers, if each peer has M short-range links, its short-range links can have C_{N-1}^{M} possible configurations. Then for the short-range links of all the peers, there exist $(C_{N-1}^{M})^{N}$ configurations. In addition, when changes happen in the network like peer joining, leaving and updating their content, the current configuration may not be the optimum anymore. Additional computation is required to recompute the optimal configuration. A classical solution for combinatorial optimization problem is using local search [Kolen 1994]. The local search approach assumes a space where all possible graph configurations reside. Each graph configuration has a set of other graph configurations as its neighbors in the space. A neighbor refers to a configuration that is reachable from the current configuration via a well-defined move. Specifically, given an initial graph configuration, local search refers to move from the current configuration to one of its neighbors. Current graph configuration is iteratively replaced by a new configuration, until the optimum configuration is achieved or a recursion bound is elapsed.

In a P2P network setting, no central point controls the local search to move the graph from one configuration to another. Instead, each peer is allowed to autonomously rewire its links to peers that are discovered in its neighborhood. So the only possible way is to let each peer optimize its local configuration (short-range links). Local optimizations move the global network configuration from one to another. If local optimizations are properly performed, a global optimal network configuration can be obtained (will be explained in Section 3.2.3). In the state of the art in SONs in unstructured P2P networks, peer rewiring is employed in this way, but it has never been framed as an optimization model, and it is not clear how peer rewiring results in the global optimization with similar peers clustered via the short-range links. In order to have a more clear idea about the relation between local optimizations and global optimization, as well as to explore better solutions to the optimization problem, we formalize the generic peer rewiring procedure into a decentralized local search solution to the optimization problem of building SONs.

In this decentralized local search solution, each peer independently performs local search to find optimal short-range contacts. Local search refers to the process in which each peer explores peers from its neighborhood, uses them to update its short-range links, and then achieves a better configuration of its short-range links. The local search is performed repeatedly until the rewiring peer achieves the optimal short-range links. By 'decentralized', we mean that each peer independently performs the rewiring process. The behaviors of all peers are expected to enable the emergence of peer clusters; By 'local', we refer to the fact that peers can only

explore peers in their neighborhood.

Similar to the optimization for the network configuration, an objective function is required for the local search operations of each peer. According to the details of peer rewiring, we define this objective function as:

$$O(p_i) = \overline{IntraDis}(p_i). \tag{3.8}$$

It corresponds to an evaluation of the local configuration of peer p_i (its short-range links). Starting from an initial configuration of short-range links that point to some random peers, p_i recursively changes its configuration to a new one with a higher value of $O(p_i)$, like moving in a configuration space. A neighbor of the current configuration is obtained by updating the current short-range contacts with peers explored in p_i's neighborhood. Exploring peers in the neighborhood is performed by a walker based on certain strategy, which is the key to the performance of this decentralized local search solution.

Formally, the local search of each peer p_i involves the following elements:

Current configuration S_i**:** p_i's current short-range links.

Configuration space CS**:** if there are N peers in the network, there are C^s_{N-1} possible configurations for each peer p_i, where s refers to the number of short-range links and is fixed.

Objective function $O(S_i)$**:** $O(S_i) = \overline{IntraDis}(p_i)$. The target of local search is to find a $S_i^{'}$ so that the objective function reaches its optimum (in the reality, the optimum may not be 0 if no such peers exist in the network). There may exists more than one optimal configuration, if there are more than s peers that are similar to p_i in the network in terms of θ.

Neighboring configuration $S_i^{'}$**:** a configuration that is reachable from the current configuration S_i. It is obtained by replacing the less similar short-range contacts with peers explored in p_i's neighborhood. Once a peer p_e is explored, a possible neighboring configuration is generated since p_e can be used to replace the least similar short-range contact in the current configuration S_i. To facilitate presenting the idea, we characterize a neighboring configuration $S_i^{'}$ by $\{p_e\}$, the explored peers that can be used to replace current short-range contacts.

With the above elements, local search of each peer is described as a periodical process, each round of the search corresponds to one *rewiring cycle*, that is composed of a sequence of operations: initiating a walker; using the walker to explore

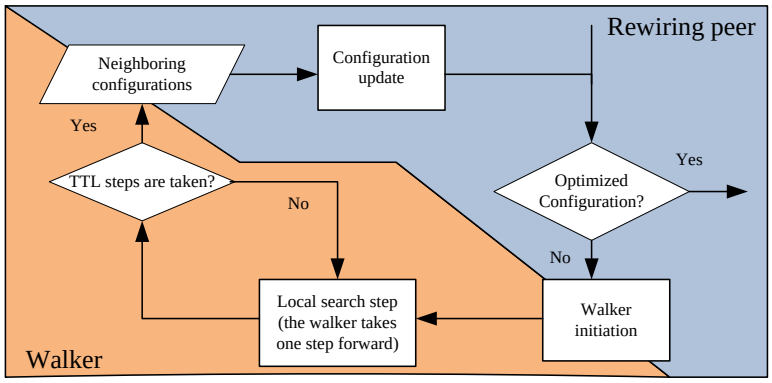


Figure 3.1: Repeated local search process of each peer (yellow part is performed by the walker to explore neighboring configuration; blue part is performed by the rewiring peer to update the current configuration with the explored ones).

the neighboring configurations; and updating the current configuration with the neighboring ones. A single step taken by the walker is considered as a single local search step for a neighboring configuration. As Figure 3.1 shows, neighboring configurations are explored with a sequence of local search steps. The peer information the walker collected at each step characterizes one neighboring configuration. All the neighboring configurations are sent back to the rewiring peer when the walker's TTL equals 0. Figure 3.2 illustrates an example about local search steps taken by a walker with TTL of 5.

The details of the local search processes are described in Algorithm 5. Specifically in each local search round, neighboring configurations are firstly explored by the walker in its neighborhood, as showed in line 5–9. TTL_{R_i} local search steps are performed in one local search round. In each local search step, the walker faces a set of possible peers to step on. These possible peers are the short-range and long-range contacts of its host peer p_r[3]. The walker has to choose one from them as its next step, called *explored peer*. LOCAL_SEARCH(R_i, p_r) in line 7 represents a local search step. It involves the strategy to select one of the contacts as the walker's next step. The information of the explored peers is recorded in the walker, and then used to update the current configuration when it is sent back to p_i (line 11–16). The neighboring configuration is accepted if only it makes any optimization over the objective function $O(p_i)$ (line 15).

The local search steps (operations of LOCAL_SEARCH(R_i, p_r)) play an im-

[3]Note the long-range contacts are also used here, since they connect up the whole network and thus similar peers may also be discovered through them, especially when the network is randomly connected.

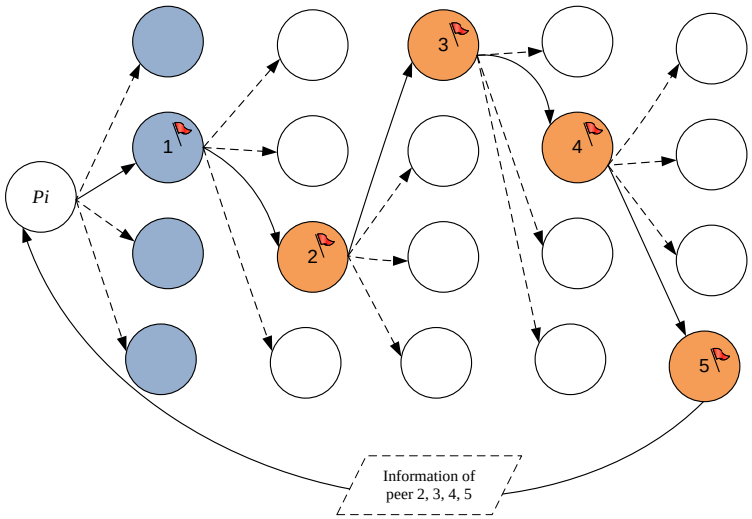


Figure 3.2: An example of local search steps performed by a walker with TTL as 5, each peer is assumed to have 4 contacts (black arrowed lines: walker's path; blue circles: p_i's current neighbors; yellow circles: explored peers).

portant role in this algorithm, because the strategy they use to select one of the contacts determines the efficiency to achieve the optimal configuration of the short-range links. It also affects the effectiveness of the resulting SON topology. This will be discussed in the next subsection.

In the classic local search approaches, the current configuration is updated once a neighboring configuration is found. But in our local search process, the current configuration is updated after TTL_{R_i} neighboring configurations are found, since the information of the neighboring configurations are returned to p_i after every TTL_{R_i} local search steps (the walker returns to p_i after its TTL is run out), and only p_i can updating its configuration. However, our local search process still can be analogized to the classic local search approach, if we consider the TTL_{R_i} neighboring configurations as a single neighboring configuration characterized by all the explored peers.

Note that a peer p_i may never achieve the optimum of its objective function due to the fact that there may not exist a set of peers in the network whose average distance to p_i is below θ. Certain mechanism must be designed to deal with this exception. For instance, the peer can stop the optimization process if its configuration is not improved in a certain number of continues local search steps. In addition,

more than one peer can be selected as the walker's next step in each local search step. In that case, the walker copies itself and each copy goes to a different peer. Moreover, $P^i_{collected}$ can collect not only the information of the explored peer, but also the information of the other contacts of the walker's host peer. These are the properties that can be specified by the algorithm designer. However, they do not affect the essential principle of this decentralized local search solution. So we will consider the basic configuration such that: peers keep optimizing their configuration; only one peer is selected as the walker's next step; only the information of the peer the walker visits is collected and returned.

Algorithm 5: Local search process of p_i

```
1  Current configuration of p_i: S_i = P^i_short ;
2  Initiate a rewiring message R_i as a walker;
3  Initiate R_i's host peer p_r as p_i ;
4  while O(S_i) < 0 do
       // explore peers (neighboring configurations)
5      P^i_collected = {} ;
6      for i = [1 : TTL_{R_i}] do
7          p_e = LOCAL_SEARCH(R_i, p_r) ;
8          P^i_collected = P^i_collected ∪ {p_e} ;
9          p_r = p_e
10     end
       // accept/refuse the neighboring configurations, each peer in
          P^i_collected implies a possible neighboring configuration
11     for p_e ∈ P^i_collected do
12         find the least similar peer p_max in S_i ;
           // form the neighboring configuration by replacing p_max
              with p_e
13         S'_i = (P^i_short − {p_max}) ∪ {p_e} ;
14         if O(S'_i) is improved then
15             S_i = S'_i;
16         end
17     end
18 end
19 return S_i;
```

3.2.3 Discussion: From Decentralized Local Search to Global Optimization

With the building of SONs modeled as a combinatorial optimization problem and peer rewiring modeled as a decentralized local optimization solution to it, an explicit difference between their objective functions can be observed. The former aims to optimize the quality of peer clusters from a global point of view, so it takes cluster efficiency into consideration in its objective function, while the latter only aims to optimize the quality of individual peers and their direct short-range contacts. There is no explicit proof showing that the decentralized local search solution always leads to the global optimization. There exists a lot of local search strategies, whose effect on this global optimization problem is still not clear. In the remaining of this subsection, we briefly analyze the link between decentralized local search and global optimization. A formal theoretical study for this problem should be conducted, which could be part of the future works.

Optimization of the Relative Intra-cluster Similarity: In a network with N peers, if each peer p_i has a set of s peers as its short-range contacts, there are C_{N-1}^{s} possible peer sets for p_i's short-range contacts. Assuming M is the number of the sets each with s peers whose average distance to p_i is below or equal to θ, the probability that a set of peers meets the criteria of short-range contacts is M/C_{N-1}^{s}. In other words, given a set of peers as p_i's short-range contacts, with the probability of M/C_{N-1}^{s}, they can optimize p_i's relative intra-cluster similarity. In unstructured P2P network, since peers can continuously rewire their links and obtain new short-range contacts, they can try a large number of the short-range contact configuration. In this case, the probability becomes higher to have the short-range contacts that can optimize peers' relative intra-cluster similarity.

Specifically, achieving the short-range contacts satisfying the criteria depends on the following factors: it is possible for a peer to access such peers in the network within the number of hops equal to the TTL of the walker for peer rewiring; the right walking strategy is designed for the walker to discover them. The former requirement can be satisfied if a proper TTL of the walker is set, since both long-range and short-range links make the whole network a connected component. The latter requirement will be studied in the next chapter, where different walking strategies are analyzed and an novel strategy is proposed.

Optimization of the Clustering Efficiency: the decentralized local search focuses only on optimizing the average intra-cluster distance between the peers and their short-range contacts. Oppositely, clustering efficiency considers the connection

between a peer p_i and all the other peers that are similar to it. This is important because the number of all the peers that are similar to p_i may be much larger than the number of its short-range links. In an optimized SON, these similar peers are supposed to be accessible from one another along the short-range links. To make sure that decentralized local search solution also results in high clustering efficiency, isolated similar peers should be avoided. Isolated similar peers refer to a set of peers with similar content, such that some of them are not accessible by following the short-range links of the others.

To avoid isolated similar peers, one way is to allow the walker to visit the contacts with dissimilar contents which may connect to a similar peer. To this end, some steps to the peers with different contents should be taken during the local search, so that a similar peer may be discovered through a step to a dissimilar peer. For example, let p_a and p_c are two similar peers that belong to two sets of isolated similar peers. They are isolated by p_b that is not similar to both p_a and p_c, which means that p_b is one of p_a's contacts and p_c is one of p_b's contacts. When p_a sends a walker to perform local search for rewiring its short-rang contacts, the walks to p_b and then p_c can help to discover p_c and then remove the isolation.

A simple way to avoid isolated similar peers is to let the walkers always take random steps, so that isolated similar peers can be gradually removed if enough amount of random steps are taken. Useless random steps can also be taken in this way, which is a waste of time and traffic. However, it is challenging to decide when to use random walk and how many random walks to use, because each peer does not know the topology of its neighborhood except its direct neighbors. In the next chapter, we propose a novel decentralized local search approach to tackle this challenge.

3.2.4 Related Work

Study on evolutionary P2P topology is not new. It is about (re)organize the links between peers to achieve a certain network topology, in order to optimize the performance of a certain task [Sakaryan 2003a]. In [Merz 2006], an evolutionary local search approach is proposed to design a P2P topology named minimum routing cost spanning trees. Given an initial topology, a local search is used to find a better topology in the neighboring topologies, which refer to a set of topologies that are reachable from the current one by a well-defined 'move'. The local search is performed by a central controller, who has the knowledge of the whole network and knows how to perform the 'moves'. Similar works are proposed in [Wolf 2009, Wiśniewski 2011]. [Ohnishi 2012] uses an evolutionary algorithm (EA) inspired by biological genetics and evolution to adapt a P2P topology for quick, accurate, and reliable searches. A

super-peer is used to manage the global knowledge of the network and perform EA in a centralized way. [Sakaryan 2003b, Sakaryan 2004] proposes to use decentralized algorithms to self-organize a static P2P network into communities each sharing similar content. It states that the algorithm is executed locally on every peer and each peer gets the information about other peers by traveling search messages. However, no detailed information is presented.

The idea of P2P topology with communities is quite similar to SONs [Crespo 2002b], where peers with similar content are 'clustered' up. It is studied in a lot of works such as [Schmitz 2004, Voulgaris 2007, Parreira 2007, Raftopoulou 2008a]. These works follow the idea of [Sakaryan 2004], and allow each peer to reconstruct its links to peers with similar content. Specifically, each peer explores the information about other peers (not its current neighbors) via gossiping [Voulgaris 2007, Parreira 2007] or traveling search messages [Schmitz 2004, Raftopoulou 2008a]. With the explored information, it reconstructs its links to peers with (more) similar content.

In this thesis, we take the current research one step further by firstly identifying a generic pattern of peer rewiring. The generic pattern is formalized as a three-step procedure that is repeatedly and independently performed by each peer. We then associate local link reconstruction with the global network evolution by modeling them as optimization models. We model the building of a SON as a combinatorial optimization problem, and peer rewiring as its decentralized local search solution. This modeling reveals an observable gap between the combinatorial combination problem and the decentralized local search solution. This motivates us finding local solutions with desirable properties that can lead to a global topology that is optimal or close to the optimum.

3.3 Summary

In this chapter, we presented a generic procedure framework for building SONs by peer rewiring. It formalizes the task of building SONs in unstructured P2P networks into a generic framework, and allows enough flexibility for the designer to specify its properties. It also allows peers' dynamic behaviors like changing content, joining or leaving the network. Then we modeled the problem of building SONs as a combinatorial optimization task, with peer rewiring as its decentralized local search solution. In the end, an analysis was made based on the optimization model, which revealed a gap between the local search solution and the global optimization task that has to be filled carefully through the identification of effective local search procedure.

Chapter 4

Simulated Annealing based Local Search for SONs

Contents

4.1 Introduction 51
4.2 Local Search: an Overview 52
4.3 Motivation 53
4.4 Simulated Annealing (SA) 55
4.5 SA-based Decentralized Local Search 58
4.5.1 Applying SA to Decentralized Local Search 58
4.5.2 Enhanced SA-based Decentralized Local Search 62
4.5.3 Cooling Schedule 64
4.5.4 Cooling Schedule with Peers' Dynamic Behaviors 67
4.6 Summary 69

4.1 Introduction

In the last chapter, building SONs based on peer rewiring is modeled as an combinatorial optimization problem with a decentralized local search solution. In this chapter, we study the possible ways to perform the decentralized local search. The local search aims to explore peers from the neighborhood of a rewiring peer p_i, and then to use them to optimize p_i's current configuration of the short-range links as well as the quality of peer clusters. It is repetitively performed, until the optimum is achieved.

The local search is autonomously performed by each peer. A single local search process of peer p_i is implemented by letting a walker walking along the links in p_i's neighborhood. The information of the visited peers is collected by the walker and returned to p_i for updating its current links. Traditional ways for a walker taking its steps include random walk, greedy walk or both [Schmitz 2004, Raftopoulou 2009a].

Random walk means the walker takes its next step randomly, while greedy walk means the walker steps to the peer which is the most similar to the rewiring peer. These strategies are individually applied or integrated in the local search process [Schmitz 2004, Parreira 2007, Raftopoulou 2009a], and are kept constant as the network evolves from a randomly connected network to a SON.

However, the evolution of p_i's neighborhood structure may affect the performance of the local search strategy. For example, before peer clusters emerge in the network, p_i's neighborhood contains random peers connected with each other. So following a link to a dissimilar peer, the walker may access a similar peer afterwards; while following a link to a similar peer, it may meet no more similar peers. As peer clusters gradually emerge, peers similar to peer p_i start to be accumulated in its neighborhood. Then following a link to a similar peer, the walker can access other similar or more similar peers with a high probability. This phenomena motivates an evolving local search strategy, which evolves from a random strategy to greedy strategy.

To this end, we propose to use Simulated Annealing (SA) to guide the walker to explore the peer's neighborhood. SA is a metaheuristic based local search approach which allows a 'bad' search being accepted with a decreasing probability. In this thesis, the 'bad' search refers to the exploration of a peer that is not useful for improving the quality of p_i's short-range contacts.

In this chapter, we will firstly present an overview of local search in Section 4.2, and then detail the idea of an evolving local search strategy in Section 4.3. Then we introduce SA in Section 4.4. We present how SA is adapted and applied in our task in Section 4.5.

4.2 Local Search: an Overview

Local search is used to find a solution among a large number of candidate solutions to optimize a quality criterion [Aarts 2003]. It moves from one solution to a neighboring one in the solution space until it arrives at the optimal solution. Neighboring solutions refer to the solutions that can be reached from the current solution according to a well-defined move, which is dependent on the specific application. The number of neighboring solutions is often much less than the total number of the solutions in the solution space. Therefore, when moving from one solution to one of its neighbors, only local information is available and considered. A good solution is the one that improves the quality criterion, while a bad solution is the one worsening the criterion.

Since local search aims at optimizing a given criterion, an intuitive way to move

from one solution to another is hill climbing. Hill climbing is an algorithm in which the initial arbitrary solution iteratively moves to a better solution selected from the local search space. It is good for finding a local optimum (the best solution among the neighboring solutions), but it is not guaranteed to find the best solution (the global optimum) out of all possible solutions (the search space). To overcome the problem of local optimum in hill climbing, the following improvement can be made: repeated local search using random restarts [Lourenço 2010]; iterated local search employing a perturbation leading to new restarts [Lourenço 2010], tabu search [Glover 1999], or simulated annealing [Kirkpatrick 1983b]. In the first two approaches, a certain number of restarts are tried in order to find the global optimum, but the local search heuristic keeps the same for each restart. In tabu search, a memory structure is used to describe the visited solutions or user-defined rules. If a potential solution has been previously visited within a certain short-term period or if it has violated a rule, it is marked as 'tabu' (forbidden) so that the algorithm does not consider it.

Simulated annealing is applied in local search approaches with the property of accepting bad solutions with a certain probability [Battiti 2009]. Comparing to the other local search approaches aiming to improve the approach of hill climbing, SA has the following features: (i) rather than restarting from a new solution when it is stuck in a local optimum using hill climbing, SA can manage to jump out of the local optimum by accepting a bad solution from the neighborhood. Through the bad solution and its neighborhood, better solution may be found. (ii) SA accepts bad solutions with a decreasing probability, controlled by a parameter called *temperature*. When the temperature is high, the probability to accept bad solutions is almost the same as the probability to accept good solutions, so the local search is similar to random walk; as the temperature goes to 0, less bad solutions are accepted, and more moves to good solutions are accepted. When this SA-based local search is applied in a combinatorial optimization problem, the gross features of the eventual status of the system appears at high temperatures thanks to the high probability to accept bad solutions, while fine details develop at low temperatures by only moving to the good solutions, as stated in [Kirkpatrick 1983a].

4.3 Motivation

We aim to efficiently build a P2P network with similar peers clustered, by letting each peer p_i to optimize the configuration of its short-range contacts as explained in Section 3.1.2. The configuration of p_i's short-range contacts is optimized when the intra-cluster distance (see Equation 3.3) is above a given threshold θ. The

optimization is performed as follows: p_i sends a walker and lets it walk along the links in p_i's neighborhood. The walker collects the information of the peers it explores for updating p_i's current short-range contacts. The exploration of a peer is considered as a local search step to find a neighboring configuration. For each explored peer, we call it good peer if the quality of p_i's intra-cluster distance is improved by replacing one of its contacts with the explored peer. Otherwise, we call it a bad peer.

The peers are explored in sequence as the walker takes its steps along the links. Their information is then returned to p_i by the walker, when its time to live (TTL) equals to 0. The good peers are then used by p_i to replace its short-range links. The walker explores the peers by following the links from one peer to another. For each step/hop it takes, the walker has more than one potential peers to step on, since each peer has more than one links. Therefore, the walker must decide which link to follow. Taking one step forward is one local search step, as formalized in Chapter 3. It is performed by either random walk or a mixture of random walk and greedy walk in the literature. In random walk, the walker takes its step by random. Random walk allows the walker randomly step to a peer that is even not similar to p_i and hence offers an extensive search. Therefore, p_i can have the chance to explore a large number of peers in the network via random walk, and then has a high possibility to achieve the optimal short-range contacts. However, the exploring process takes time because some random steps may end up without finding any good peers. So a combination of random walk and greedy walk is often used as a compromise [Schmitz 2004, Voulgaris 2007, Raftopoulou 2008a, Raftopoulou 2009a], and each is employed with a fixed probability; in [Raftopoulou 2009a], the author also proposed to perform random walk when p_i's intra-cluster distance is above a threshold and greedy walk when it is below a threshold. But this strategy does not significantly outperforms the other strategies.

However, as the local search drives the network gradually towards a SON, the evolution of the network (specifically the neighborhood structure) may affect the performance of the local search strategy. For example, before peer clusters emerge in the network, p_i's neighborhood consists of random peers connected with each other. So following a link to a dissimilar peer, the walker may access a similar peer afterwards; while following a link to a similar peer, it may meet no more similar peers. As peer clusters gradually emerge, peers similar to peer p_i start to be accumulated in its neighborhood. Then following a link to a similar peer, the walker can access other similar or more similar peers with a high possibility. This phenomena motivates an evolving local search strategy, which evolves from a random strategy to greedy strategy.

In this thesis, the local search strategy is considered in an evolving way. According to our observation, as peers perform repeated local search process, the structure of p_i's neighborhood gradually changes: the number of random peers decreases and the number of similar peers increases. In the beginning, random walk may be the best way to explore good peers. Gradually, the walker can access more and more good peers by following a link to a good peer. So it can be better if the probability to use different local search strategies can be gradually controlled. In the controlled local search strategy, random walk is used with a high probability in the beginning. As the neighborhood structure evolves, less random walk is taken while more greedy walk is performed. To this end, SA can be used because it can accept bad solutions with a decreasing probability, and thus a evolving local search strategy is possible to be implemented. The other local search approaches like iterative local search and tabu search do not have this property.

4.4 Simulated Annealing (SA)

SA originally comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to a low energy state, which refers to a highly ordered state such as a crystal lattice. To accomplish this, the material is heated into a temperature that allows many atomic rearrangements. The material is then cooled carefully until it freezes into a good crystal. This notion of cooling is employed in the Simulated Annealing algorithm to solve optimization problems. It conducts a poor solution to iteratively move to another solution, until an optimal solution (or a solution close to the optimum) is reached [Rutenbar 1989].

To implement SA in an optimization task, a sequence of local search steps are performed until the optimum is achieved. An objective function is defined to quantify the fitness of the solutions. At each single search step, SA randomly picks up one of the neighboring solutions $S^{'}$ of the current solution S, and probabilistically decides between moving to $S^{'}$ or staying at S. If it decides to move to the neighboring solution $S^{'}$, the current solution is updated as $S^{'}$. This step is repeated until the solution is optimal according to a certain criterion, or until a given computation budget has been exhausted. Following the description in [Henderson 2003], we present a detailed SA-based maximizing process in Algorithm 6.

SA starts from an initial solution S_0 and keeps moving to other solutions either until $tMax$ steps are performed or an optimal solution is found. The optimal solution is quantified as the optimal value $eMax$ of its objective function. The value of the objective function is called *energy*, calculated by calling ENERGY(). The call NEIGHBOR(S) (line 10) generates a neighboring solution $S^{'}$ of current

solution S; the call P(E, $E^{'}$, T_t) returns a probability. If the probability is larger than the value returned by RANDOM() (line 10), we move the current solution E to the neighboring solution $S^{'}$ (line 11). This is also mentioned as the neighboring solution $S^{'}$ is accepted with the probability P(E, $E^{'}$, T_t).

Algorithm 6: Simulated Annealing (SA)

```
1  S = S_0; // current solution
2  E = ENERGY(S); // current energy
3  relaxationTime = 0; // counter for relaxation time
4  t = 0; // current step of cooling schedule
5  while t < tMax and E < eMax do
6      T_t = TEMPERATURE(t); // T_t = T_0, if t = 0
7      while relaxationTime < rTime do
           // rTime is the computation budget
8          S' = NEIGHBOR(S);
9          E' = ENERGY(S');
10         if P(E, E', T_t) > RANDOM() then
11             S = S'; E = E'; // move the current solution to the
                   neighboring one
12         end
13         relaxationTime = relaxationTime + 1;
14     end
15     t = t + 1;
16 end
17 return S
```

The probability to accept a neighboring solution is controlled by the energy of the current solution E, the energy of the neighboring solution $E^{'}$, and the current temperature T_t. In traditional SA algorithm, this is implemented by Metropolis dynamics [Metropolis 1953], which always accepts the good solution, and accepts the bad solution with a given probability between 0 and 1. Consider a minimizing problem, Metropolis dynamics can be formally described in the following equation:

$$P(E, E^{'}, T_t) = \begin{cases} 1, \text{ if } E^{'} \leq E \\ e^{-(E^{'}-E)/T_t}, \text{ if } E^{'} > E \end{cases} \tag{4.1}$$

The process that Metropolis dynamics is repeatedly used to accept/refuse neighboring solutions can be regarded as simulating a Markov chain [Bertsimas 1993].

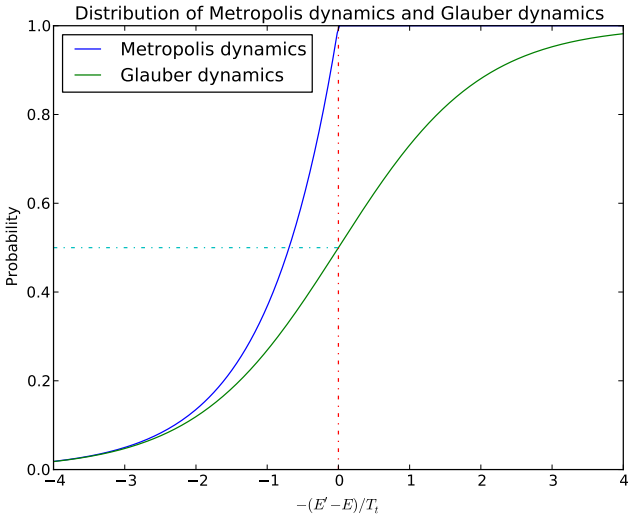


Figure 4.1: Probabilistic distribution of Metropolis and Glauber dynamics.

Similar to Metropolis dynamics, Glauber dynamics can also be used to accept/refuse a state transition in a system with multiple elements, and to simulate a Markov chain [Levin 2009]. Consider a minimizing problem, Glauber dynamics accepts both good and bad solutions with a probability defined as follows:

$$P(E, E^{'}, T_t) = \frac{1}{1 + e^{(E'-E)/T_t}}. \tag{4.2}$$

Figure 4.1 shows, for a given value of $-(E^{'} - E)/T_t$, the probability distribution of Metropolis dynamics and Glauber dynamics.

T_t is a parameter in analogy with the temperature in physical annealing. In optimization problems, however, the temperature is rather a controlling parameter to drive the initial solution to the optimum [Kirkpatrick 1983a]. Starting from a heuristic high value, the temperature is controlled by a *cooling schedule*. The cooling schedule determines when the current temperature should be lowered, and how much it should be lowered. In Algorithm 6, the former is implemented by a *relaxation time* $rTime$, which refers to the number of attempts of local search that are performed before the temperature is lowered; the latter is implemented by the call TEMPERATURE(), which yields a lower temperature given the current annealing step t.

A high temperature tends to allow more bad moves from current solution, while a low temperature allows less. Similar to the physical annealing process, cooling schedule gradually decreases the temperature T_t as the system approaches to its optimal solution. An effective cooling schedule is essential to reduce the

amount of time required to find an optimal solution. However, how the temperature is specifically scheduled often depends on the specific optimization problem [Bertsimas 1993]. The commonly used cooling schedules include exponential schedule in Equation 4.3, linear schedule in Equation 4.4, and logarithmic schedule in Equation 4.5 [Nourani 1998].

$$T_t = T_0\, a^t \qquad 0 < a < 1,\, t > 0 \tag{4.3}$$

$$T_t = T_0 - a \times t \tag{4.4}$$

$$T_t = \frac{a}{\log\,(t+1)} \qquad t > 0 \tag{4.5}$$

4.5 SA-based Decentralized Local Search

Although SA has been applied in the local search approach for the optimization problems, applying SA in the decentralized local search in this thesis is not trivial. First of all, the local search mechanism in this thesis is not exactly the same to the traditional local search. It allows a flexible local search strategy, but does not move to the other solutions immediately. Secondly, we must formalize the definition of solution and energy in our task so that SA can be applied. Moreover, the P2P system we aim to optimize has a variable size and state because of peers' dynamics behaviors. In this section, we will present how the SA is adapted into our task.

4.5.1 Applying SA to Decentralized Local Search

SA is applied to guide a walker exploring p_i's neighborhood. Specifically, the walker takes its next step to a good or bad peer by a probability controlled by SA. A good peer is defined as the peer whose similarity to p_i is higher than p_{max}, the peer that is p_i's short-range contact that has the least similarity (also mentioned as the biggest distance) to p_i. So by replacing the peer p_{max} with the good peer, the intra-cluster distance is decreased between p_i and its short-range contacts; a bad peer is defined as the peer whose similarity to p_i is not higher than the similarity between p_i and p_{max}. According to the principle of SA, at a high temperature, bad peers are accepted as the walker's next step with high probability; as the temperature decreases, the probability to accept bad peers decreases, thus more good peers are accepted. The temperature decreasing process corresponds to the evolution of peer's neighborhood structure. A higher temperature implies a neighborhood structure with a lot random connections between random peers, as the temperature decreases,

similar peers emerge in the neighborhood and gradually connect with each other. So at a high temperature, a step to a bad peer is helpful to discover good peers in the next step. While as the neighborhood structure evolves, a step to a good peer is more helpful to discover good peers in the next step.

We make a formal analogy between SA and our task in Table 4.1. The aim of our task is to search good peers to replace a peer p_i's current short-range contacts. So when a walker decides to step on a peer $p_r^{'}$, the current solution refers to the current configuration of peer p_i's short-range links. A new solution refers to a new configuration where the most distant short-range contact p_{max} is replaced by $p_r^{'}$. Let peer p_r is the host peer of the walker R_i, a possible new solution can be generated by taking any of p_r's neighbors as R_i's next step. The energy of a solution is measured by intra-cluster distance, the average distance between p_i and the short-range contacts in the configuration.

Table 4.1: Exploring a peer p_i's neighborhood with SA

A solution	The configuration of p_i's short-range contacts
Current solution	S = p_i's current short-range contacts P^i_{short}
Current energy	$E = IntraDis(p_i)$
Neighboring solution	$(P^i_{short} - \{p_{max}\}) \cup \{p_r^{'}\}$, $p_r^{'}$ is R_i's next stop, p_{max} is p_i's least similar short-range contact
Energy of neighboring solution	$\frac{1}{s}\sum_{p_k \in (P^i_{short} - \{p_{max}\}) \cup \{p_r^{'}\}} Dis(p_k, p_i)$
Time budget	$tMax$
Relaxation time	TTL of the walker R_i

So the goal becomes to minimize the solution until its energy achieve the intra-cluster distance threshold θ. The probability to accept a neighbor solution is defined by Metropolis or Glauber dynamics. As stated previously, exploring peer's neighborhood is a repeated local search process. The temperature is cooling down as peer's neighborhood is repeatedly explored. In a single local search iteration, the walker continuously takes a number of steps. The number of steps is equal to the walker's TTL, which can be specified by the system designer. To calculate the probability of taking $p_r^{'}$ as the next step, the energy difference ΔE and current temperature are required. The latter is controlled by cooling schedule, and the former is calculated by Equation 4.6. Since the number of the short-range links s is kept the same, the equation can be simplified as $\Delta E = Dis(p_r^{'}, p_i) - Dis(p_{max}, p_i)$:

$$\begin{aligned}\Delta E &= E^{'} - E \\ &= \frac{1}{s}\sum\nolimits_{p_k \in (P^i_{short} - \{p_{max}\}) \cup \{p^{'}_r\}} Dis(p_k, p_i) - IntraDis(p_i) \\ &= \frac{1}{s}\sum\nolimits_{p_k \in (P^i_{short} - \{p_{max}\}) \cup \{p^{'}_r\}} Dis(p_k, p_i) - \frac{1}{s}\sum\nolimits_{p_k \in P^i_{short}} Dis(p_k, p_i) \\ &= \frac{1}{s}(Dis(p^{'}_r, p_i) - Dis(p_{max}, p_i))\end{aligned} \tag{4.6}$$

Algorithm 7: Local search process of p_i

```
1  t = 0; // initial time
2  T_t = T_0; // initial temperature
3  while IntraDis(p_i) > θ or t < tMax do
4      Initiating the walker: R_i = ⟨IP(p_i), Prof_{p_i}, TTL_{R_i}, d_max, C⟩;
5      p_r = p_i; // p_r is the current host peer of the walker
       // R_i explores the neighborhood by a sequence of local search
          steps
6      while TTL_{R_i} > 0 do
           // search new host peer for next step
7          p'_r = LOCAL_SEARCH(R_i, p_r, T_t);
           // record the information of the new host(explored) peer
8          Append ⟨IP_{p'_r}, Dis(p'_r, p_i)⟩ to R_i.C;
9          p_r = p'_r; // p'_r becomes the current host peer
10         TTL_{R_i} = TTL_{R_i} − 1;
11     end
12     R_i returns to p_i;
13     Update short-range contacts of p_i using Algorithm 1;
14     t = t + 1;
15     T_t = TEMPERATURE(t);
16 end
```

Algorithm 7 details the repeated local search algorithm using SA: p_i periodically starts local search process until the optimum is achieved or a time budget is finished. In each local search iteration, a number of local search steps are performed sequently. The searched solutions are returned to p_i, and p_i decides to stay with the current solution or move to the other solutions. This corresponds to the exploring behaviors of the walker in p_i's neighborhood. It is implemented in line 4–11. New solutions featured by $p^{'}_r$ are collected by the walker (by storing the information of $p^{'}_r$ in R_i).

$p_r^{'}$ is selected from the neighbors of the walker's current host peer p_r, based on SA. This implementation is described in Algorithm 8.

Algorithm 8: LOCAL_SEARCH(R_i, p_r, T_t)

```
Input: R_i, p_r, T_t; // the walk, the walker's current host peer, and
    the current temperature
checkedPeer = {}; // record the searched peers
while p_r': randomly selected from (P^r_short ∪ P^r_long) − checkedPeer do
    // randomly try the unchecked peers in the local search space
        P^r_short ∪ P^r_long
    checkedPeer = checkedPeer ∪ {p_r'};
    if P(R_i.d_max, Dis(p_r', p_i), T_t) > RANDOM() then
        return p_r';
// if no peer is accepted as the walker's next step, chose one
    randomly
p_r': randomly selected from P^r_short;
return p_r';
```

Current solution is not changed until a set of new solutions are collected by the walker. This is in accordance with the process of peer rewiring described in Chapter 3, where peer p_i does not update its neighbors until the walker finishes exploring the neighborhood. It updates them when it finally receives the returned walker which carries the information of the new solutions. Note that this does not affect the fact that bad peers (bad solution) can be accepted as the walker's next step and an extensive local search can be performed, because the peers are explored in sequence and accepting bad peers may result in a set of good peers as the walker's future steps. The number of local search steps is the TTL of the walker.

T_t is used to control the exploration where local search is performed. With the returned explored peers whose information is stored in R_i, p_i updates its short-range contacts. Long-range contacts can also be updated in the meanwhile, but it is an operation specified by the protocol designer as stated in Chapter 3. If the energy of discovered solutions is not better than the current energy, p_i keeps its current short-range links, and the energy of current solution does not change.

The temperature during a single search process remains the same. This process is then repeated with a decreased temperature generated by TEMPERATURE(t), until the intra-cluster distance is optimized to be equal to or below θ.

Peers are accepted as the walker's next step according to the probability decided

by either Metropolis or Glauber dynamics, as showed in lines 5–6 in Algorithm 8. The probability is calculated based on two inputs: the current temperature and the energy difference between the current solution and the new solution.

Since the energy difference should be calculated when a walker selects a peer as its next step, the rewiring message carried by the walker has to include the information of the distance between its initiator p_i and the least similar distant short-range contact $p_{max} = Dis(p_{max}, p_i)$, represented as d_{max}. The rewiring message is then reformatted as a tuple $\langle IP_{p_i}, Prof_{p_i}, TTL_{R_i}, d_{max}, C\rangle$. If no peer is selected as the walker's next step by Metropolis or Glauber dynamics, a random one is pick instead (line 8).

4.5.2 Enhanced SA-based Decentralized Local Search

In the previous subsection, we apply SA in the local search task for peer rewiring without changing any detail of the task. We use the probability controlled by SA to select a peer as the walker's next step. The selected peer corresponds to a new solution, which is not used to replace the current solution immediately but kept in the rewiring message until it returns to the rewiring peer. This is not exactly what traditional SA does in local search, where the new solution replaces the current one once it is accepted.

In addition, the walker only collects the information of the peer that is selected as its next step, even thought the whole local search space can be easily accessed, since each peer has a good knowledge about its neighbors. Instead of only collecting the information of the walker's next step, the walker can also collect the information of the other good solutions in the current local search space, which may accelerate the optimization process.

In this subsection, we propose an enhanced algorithm (Algorithm 9) to the previous algorithm (Algorithm 7). Instead of keeping the current solution unchanged during a single round of exploration, we propose to perform solution transition immediately after each single step of the walker. To implement this, additional information has to be included in the rewiring message R_i. Specifically, we append to R_i a ranked list of distance values between p_i and its short-range contacts. After each walk step to $p_r^{'}$, the biggest distance value is replaced by the distance between p_i and $p_r^{'}$, if the latter is smaller than the former. The replacement can be regarded as removing an old short-range link and building a new one (although the real replacing operation will take place after R_i returns to p_i). By doing this, the current solution is updated in real time, which may be helpful to the subsequent exploration process. In addition, except collecting the information of the peer $p_r^{'}$ (the next step of the

walker R_i), the walker is also allowed to collect the information of the other good solutions in the local search space. These additional good solutions are also returned to p_i in order to speed up the optimization process of its short-range links.

Algorithm 9 describes the details of this enhanced SA-based local search. The real-time update for the solution is performed in lines 6–8. Lines 10–14 show how the walker collects the information of other good solutions.

Algorithm 9: ENHANCED_EXPLORATION(p_i, T_t)

1 $rankedDis = \text{SORT}(Dis(p_j, p_i), \forall p_j \in P^i_{short})$; // generate the ordered distance list
2 $R_i = \langle IP(p_i), Prof_{p_i}, TTL_{R_i}, rankedDis, C \rangle$; // initiate the walker, containing $rankedDis$
3 $p_r = p_i$; // initiate the host peer of the walker
4 **while** $TTL_{R_i} > 0$ **do**
5 $\quad p_r^{'} = \text{LOCAL\_SEARCH}(R_i, p_r, T_t)$;
$\quad$ // updating $rankedDis$ in R_i if $Dis(p_r^{'}, p_i)$ is smaller than one of the elements in $rankedDis$
6 $\quad$ **if** $Dis(p_r^{'}, p_i) < MAX(R_i.rankedDis)$ **then**
7 $\quad\quad R_i.rankedDis = (R_i.rankedDis - \{MAX(R_i.rankedDis)\}) \cup \{Dis(p_r^{'}, p_i)\}$;
8 $\quad$ **end**
$\quad$ // collect information of $p_r^{'}$ in R_i
9 $\quad$ Append $\langle IP_{p_r^{'}}, Dis(p_r^{'}, p_i) \rangle$ to $R_i.C$;
$\quad$ // continuing appending p_r's other short-range contacts to R_i if they are good peers
10 $\quad$ **for** p_j *in* $P^r_{short} - \{p_r^{'}\}$ **do**
11 $\quad\quad$ **if** $Dis(p_j, p_i) < MAX(R_i.rankedDis)$ **then**
12 $\quad\quad\quad$ Append $\langle IP_{p_j}, Dis(p_j, p_i) \rangle$ to $R_i.C$;
13 $\quad\quad$ **end**
14 $\quad$ **end**
15 $\quad p_r = p_r^{'}$;
16 $\quad TTL_{R_i} = TTL_{R_i} - 1$;
17 **end**
18 **return** R_i

4.5.3 Cooling Schedule

A cooling schedule is used to control the cooling rate of the temperature. In physical annealing process, it is assumed that cooling rate should be low enough for letting the probability distribution of the system state to be near thermodynamic equilibrium [Bertsimas 1993]. A relaxation time is defined to indicate the time one must wait for the equilibrium to be restored after the change of temperature. It strongly depends on the 'topography' of the energy function and on current temperature. In SA algorithm, these configurations about cooling schedule do not have the exact meaning as that in physical annealing process [Brooks 1995], they are rather controlling parameters that are often achieved by empirical study. Adaptive simulated annealing algorithms [Ingber 2012] are proposed to automate the cooling schedule by connecting it to the search progress. However, we are more interested in observing the performance of SA-based local search given an appropriate cooling schedule. Hence, in this thesis, the appropriate cooling schedule is set by trial-and-error in a preliminary experiment: the cooling schedule that achieves the best local search result is used for our study.

Some heuristics can be used to choose the appropriate cooling schedule function as well as the relaxation time at any temperature. We consider to keep the same temperature while the walker is sent to explore the neighborhood, and decrease it when another walker is sent to perform another iteration of neighborhood exploration. For each temperature, the relaxation time is set as the TTL of a walker, which refers to the number of the attempts of local search. During the time a peer performs the local search to update its short-range contacts (the current solution), other peers in the network must also updates their configurations of the short-range contacts considering that peers rewire their connections in parallel. The updated configurations change the topology of network. Therefore, the temperature must change between any two iteration of local search in order to reflect the change of the system. However, the temperature in a single iteration of local search should remain the same, because no neighboring solution is returned to the rewiring peer and allows it to update its current solution before the TTL of the walker reaches 0, and thus it can be assumed that no significant changes happen in the network topology during this period of time.

For choosing the proper cooling schedule function, we need to consider the effect of temperatures on the energy differences. A proper cooling schedule should be able to control the temperature so that the probability to accept any solution has the intrinsic principles of SA [Brooks 1995]: (i) the probability is almost the same to accept any bad solution in the beginning of the cooling schedule, no matter how much the resulting energy difference is; (ii) as temperature decreased, the value of

the energy difference must also effect the probability. For example, the probability to accept a bad solution with energy difference 0.1 must be smaller than the probability to accept a bad solution with energy difference 0.5.

Now let's consider how the cooling schedule can respect the above principle in our task. In our task, since the Jaccard distance between two peers ranges between 0 and 1, the absolute value of the energy difference achieved in each local search step is also in the same range. In Tables 4.2 and 4.3, the probability to accept a solution is listed given a range of temperature and a certain value of energy difference for Metropolis and Glauber dynamics, respectively.

Table 4.2: The probability of Metropolis dynamics given the value of the temperature and the energy difference

		$\triangle E$				
		0.2	0.4	0.6	0.8	1.0
T_t	0.5	0.67	0.44	0.30	0.20	0.13
	1	0.81	0.67	0.54	0.44	0.36
	2	0.90	0.81	0.74	0.67	0.60
	10	0.98	0.96	0.94	0.92	0.90
	50	0.99	0.99	0.98	0.98	0.98

Table 4.3: The probability of Glauber dynamics given the value of the temperature and the energy difference

		$\triangle E$									
		-1.0	-0.8	-0.6	-0.4	-0.2	0.2	0.4	0.6	0.8	1.0
T_t	0.5	0.88	0.83	0.76	0.68	0.59	0.40	0.31	0.23	0.16	0.11
	1	0.73	0.68	0.64	0.59	0.54	0.45	0.40	0.35	0.31	0.26
	2	0.62	0.59	0.57	0.54	0.52	0.47	0.45	0.42	0.40	0.37
	10	0.52	0.52	0.51	0.51	0.50	0.49	0.49	0.48	0.48	0.47
	50	0.50	0.50	0.50	0.50	0.50	0.49	0.49	0.49	0.49	0.49

Since the probability distributions of Metropolis and Glauber dynamics are monotonically increasing (Figure 4.1), we can observe, given a range of energy difference [-0.2, -1.0], any temperature above 10 generates the probability ranges [0.90, 1) for accepting bad solution in Metropolis dynamics, and [0.47, 0.5) for accepting bad solution in Glauber dynamics. A wider range of probability can only be achieved with a temperature smaller than 2. For example, when the temperature is 0.5, with

Metropolis dynamics, an energy difference -0.2 corresponds to a probability 0.67, while an energy difference -0.1 corresponds to a probability 0.13.

Assuming that the cooling schedule starts from a quite high initial temperature, like 500, we can observe that it works almost the same when the scheduled temperature is above 10, since the temperature above 10 allows to accept any bad solutions with a probability close to 1 (Metropolis dynamics) or 0.5 (Glauber dynamics). Local search in this case is like random walk. When the temperature is cooled to be below 2, the probability to take bad solutions starts to decrease. When the temperature becomes very low, the probability to take bad solutions is close to 0. Particularly in Glauber dynamics, when the temperature decreases, the probability to take good solutions also changes. It gradually increases as the temperature decreases. When the temperature becomes very low, the probability to take good solutions is close to 1. Local search in this case becomes greedy walk, so that only good solutions are accepted.

In order to allow a gradual evolution of the local search strategy (from random to greedy) with Metropolis or Glauber dynamics, the cooling rate should be carefully controlled to avoid too many random walks or too many greedy walks. For example, with the linear cooling schedule showed in Figure 4.2, the temperature remains high for a long time, allowing a large number of random walks; as the temperature becomes low, it continuously decreases with the same rate, which only allows a small number of greedy walks before it reaches 0. Instead, with the logarithmic cooling schedule, the temperature decreases very rapidly to the temperature which allows greedy walks. So there are not enough random walks to extensively explore the neighborhood to allow the peer clusters to emerge. If we consider a cooling schedule which allows a high cooling rate when the temperature is above 2 and a slow cooling rate when the temperature is below 2, some useless random walks can be avoided, and greedy walks are gradually allowed. Good peers can be discovered more efficiently in this way. So both of these two cooling schedules are not appropriate for our task. Only the exponential cooling schedule is in accord with our requirement, which starts from a high temperature, and has a proper cooling speed in the beginning and slows down after the temperature become low.

As a conclusion, we give a general picture about SA-based local search for all peers in unstructured P2P network. From unstructured network to SONs, P2P network gradually changes its topology. The initial random links are gradually changed to be links between similar peers. This is analogized to a cooling process from a high temperature to a low one. For each peer in the network, at a high temperature, it explores the neighborhood almost randomly; as the temperature decreases, it tends to guide the walker to the good peers with a higher probability.

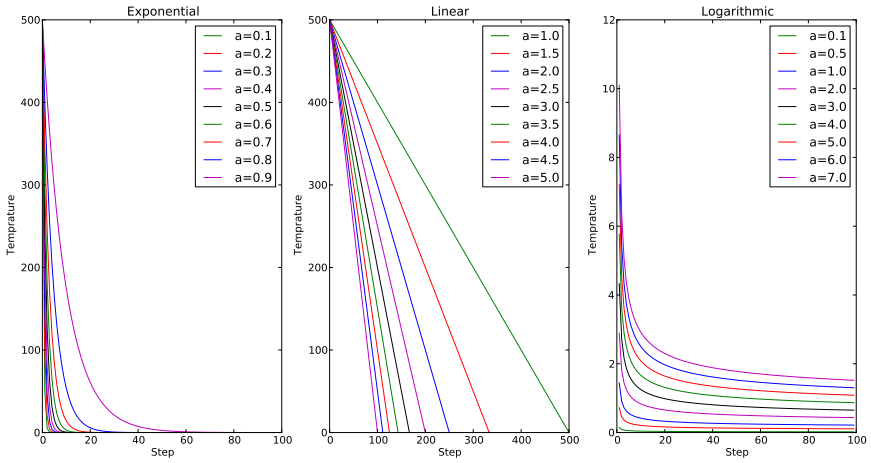


Figure 4.2: Cooling schedule with initial temperature $T_0 = 500$.

Given a random P2P network, peers perform SA-based peer rewiring independently until their intra-cluster distance reaches the threshold θ. The cooling schedule for SA-based peer rewiring is set as Equation 4.3. The same initial temperature and same cooling rate are set for each peer, considering that they reside in the same system.

4.5.4 Cooling Schedule with Peers' Dynamic Behaviors

The P2P network we study consists of dynamic behaviors like peer joining/leaving and content updating. In Figure 4.3, we demonstrate the possible dynamic behaviors of peers during the process of network topology evolution. These dynamics behaviors result in new situations for peers rewiring.

The first situation is caused by new peers joining the network during the network evolution. SA-based peer rewiring of these new peers has to be treated differently, because their initial temperature depends on the state of the network when they join. For example, if the peer clusters already emerge in the network, the new peer does not need a high initial temperature to randomly explore its neighborhood; if the peers are still randomly connected with each other when a new peer join the network, it should start peer rewiring at a high temperature, because random walk is necessary in this case to explore possible good peers in its neighborhood. Therefore, given a new peer joining the network during the network evolution, a reasonable initial temperature should be set by considering the network state.

The second situation is that existing links that met the requirement can be changed. For example, a short-range contact may leave the network; or it may

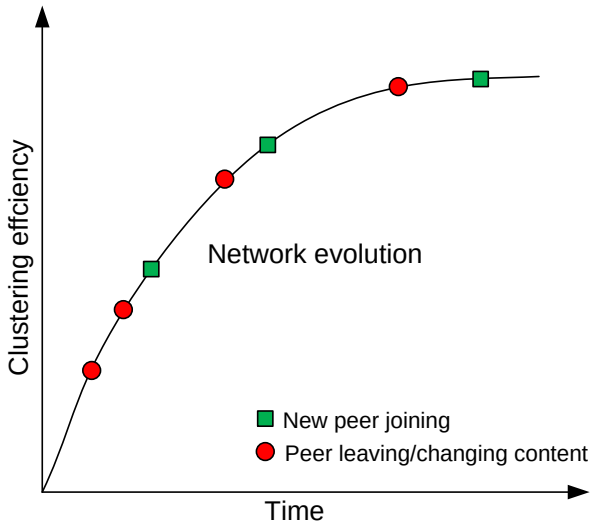


Figure 4.3: Peers' dynamic behaviors can happen at any time during the network evolution, in which the clustering efficiency is assumed to increase monotonically in this illustration.

change its content and accordingly change its profile. This may cause the intra-cluster distance above the threshold θ. In this case, peers have to restart peer rewiring to rewire the links. SA-based peer rewiring in this case also needs an appropriate initial temperature according to the network topology of that moment.

Therefore, a consistent cooling schedule schema is proposed in Algorithm 10. It integrates peers' dynamic behaviors in the process of generating SONs from a random P2P network. In this algorithm, the initial temperature for a new peer is decided by the temperature of its neighbors, which implies the situation of current network topology. Specifically, it is calculated as the average temperature of the peers's neighbors, which includes its long-range contacts and short-range contacts (line 11). The traditional cooling schedule decreases the temperature monotonically, and the probability to accept bad solutions also decreases. When the temperature is close to 0, only good solutions are accepted, since a temperature close to 0 also results in the probability to accept bad solutions close to 0 (line 9). Cooling the temperature in this way works perfectly for generating SONs when no dynamic behaviors exist in the network, because as the similar peers are gradually clustered, stepping on good peers can explore more good/better peers, and the necessity to step on bad peers to explore good peers decreases.

Algorithm 10: TEMPERATURE(t,p_i)

```
1  G =< P, L >;
2  if p_i in G.P then
3      if t == 0 then
4          return T_0; // initially, the temperature is the same for all
               the peers that originally exist in the network
5      else
6          if IntraDis(p_i) is not improved and no bad peers are accepted in the
           last ξ local search iterations then
7              T_t = RESET();
8          else
9              T_t = T_0 a^t;
10 if p_i is not in G.P then
11     T_t = 1/(s+l) Σ_{p_j ∈ P^i_short ∪ P^i_long} T_{t,p_j}; // T_{t,p_j} is the temperature of p_j, p_j
           is one of p_i's s + l contacts
12 return T_t
```

However, if the dynamics in the network happen when the temperature is very low and it involves a large number of peers, the network must be reconfigured into an optimum state. To this end, certain bad peers should be accepted when the walker explores the neighborhood in order to reconnected the isolated similar peers. But with a low temperature close to 0, bad peers are not possible to be accepted as the walker's next step. Therefore, the temperature must be reset to be a high value in order to allow bad peers to be accepted. Formally, in order to decide when to reset the temperature, certain information has to be recorded in the peer's last ξ local search iterations: the improvement the peer achieves over its intra-cluster distance and the number of bad peers that are accepted. If the peer's intra-cluster distance has not been improved and no bad peers are accepted in the last ξ local search iterations (line 6), we follow the idea from [Battiti 2009], to increase the temperature by a heating factor h larger than one at each iteration during the processing of reheating (line 7).

4.6 Summary

In this chapter, a SA-based approach was introduced in the decentralized local search task for building SONs. Firstly, we presented our motivation to use SA

instead of other local search approaches. Then we recast SA into our task, with each peer performing a SA-based local search algorithm to optimize the configuration of its short-range links. An appropriate cooling schedule was then selected for the proposed approach. We concluded the chapter with the proposal of a dynamic cooling schedule, which adaptively sets the peers' temperature according to the situation of their neighborhood that is affected by the network evolution as well as the dynamic behaviors of peers.

Chapter 5

Evaluation

Contents

5.1 Data Preparation 72
5.1.1 Data Requirement 72
5.1.2 Related Work 72
5.1.3 Using Reuters Corpus 75
5.1.4 Generating Peer Profile 75
5.2 Experimental Setup 76
5.2.1 Configuring the Network: Setting Parameters 76
5.2.2 Simulation Procedure 78
5.3 Metrics 80
5.4 Results and Analysis 81
5.4.1 Building SONs from Random Networks 81
5.4.2 Information Retrieval in SONs 94
5.4.3 Enhanced SA-based Local Search 99
5.4.4 Building SONs with Dynamic Behaviors: Peer Joining 103
5.4.5 Configuration VS. Performance 104
5.5 Summary 114

In this chapter, we evaluate our approach, Simulated Annealing (SA) based decentralized local search approach, by simulating it to build Semantic Overlay Networks (SONs). Firstly, we introduce how we prepare the content in peers for the simulation (Section 5.1). The properties of the required data are identified, and a proper data set is chosen based on an overview of the related work. The data set is distributed in the peers of a simulated P2P network, according to the category information of its documents. Peer profiles are then calculated. In Section 5.2, we detail the configuration of the P2P network connections and our approach. We then present the simulation procedure, from which, the results are evaluated by the metrics introduced in Section 5.3, and presented and analyzed in Section 5.4. We finally summary our simulation results in Section 5.5.

5.1 Data Preparation

There exist many data sets for evaluating Information Retrieval (IR) in SONs. This rises the problem of how to properly choose the data set. We firstly identify the properties of the data set required in this thesis, then review the related works about data preparation. Based on the review, Reuters corpus [corpus 2011] is chosen for evaluating our proposal. Details are then described about how Reuters corpus is distributed to peers, how each peer calculates its profile, and how queries and reference assessment are achieved.

5.1.1 Data Requirement

We identify 3 properties of the data that is required for evaluating our approach:

- **Property 1**: The data should be a collection of text documents over which text-based IR can be performed, since we build SONs for the application of full-text P2P-IR. It should have queries and relevance assessment in order to evaluate the performance of IR.
- **Property 2**: The documents in each peer is assumed to represent the user's interests. So additional information like topics or categories about the documents should be available or easily obtained, in order to facilitate distributing the documents to peers.
- **Property 3**: The document collection should be large enough to be distributed to a P2P network with a large number of peers, so that our approach can be evaluated in not only the networks with small size but also the networks with large size.

5.1.2 Related Work

There are mainly four sources of data sets for IR in P2P networks. One of the resources is from TREC [1], which provides data collections for evaluating centralized text Information Retrieval. For IR in P2P networks, TREC 5, 6 and 9 are commonly used. For example, TREC 5 is used in [Balke 2005]. In this work, documents were randomly distributed to peers, and queries with 2 words on average were randomly generated from the documents. The P2P-IR results are compared to a ground truth, which is generated by evaluating the same queries against the whole TREC 5 document collection in a centralized way. TREC 6 [2] and TREC 9 [3] are used

[1] http://trec.nist.gov/
[2] http://boston.lti.cs.cmu.edu/callan/Data/
[3] http://trec.nist.gov/data/t9_filtering.htm

in [Raftopoulou 2008a]. Based on the category information of the corpus, each peer keeps a set of documents in one category. This corresponds to the assumption that each peer has one interest. ClueWeb12 data set [4] is another data set published by TREC. It is comprised of roughly 1 billion web pages. It is also used for IR in P2P networks [Ke 2010]. In this work, the web pages belonging to the same web site were distributed to one peer.

Free resources like Wikipedia[5], MEDLINE[6] and Reuters [Lewis 2004] have been also used. In [Podnar 2007, Skobeltsyn 2006], the authors crawled Wikipedia documents and distributed them randomly over peers. They used the real query logs of Wikipedia for IR in P2P networks. The ground truth relevant documents were generated by a centralized search engine using the BM25 relevance schema[7]. In [Papapetrou 2010], MEDLINE and Reuters were used. The authors deploy documents to peers based on the category information of documents. Reuters-21578, a sub collection of Reuters, was used in [Doulkeridis 2010].

In the works dealing with the general task of IR in P2P networks, real text documents are not used for the evaluation. Instead, artificial data is often produced to validate the proposed approaches, such as in [Crespo 2002a, Kumar 2005a, Marin 2009, Kim 2011].

Another alternative of data set is real traces from P2P file sharing systems or Online Social Networks (OSN). The latter is becoming a main data source for IR in P2P networks, due to the recently emerging research in social networks. For example, in [Bender 2007], the authors crawled a part of del.icio.us[8], and assume each peer corresponds to a user. The documents in each peer were the bookmarks the user shared. The tags annotated to the bookmarks were used as queries, and the ground truth relevant documents are the bookmarks that contain the same tags. Real traces crawled from CiteULike [9] was used in the similar way in [Bertier 2010].

We summarize the data sets used in the literature in Table 5.1. Except TREC, MEDLINE, Reuters and ClueWeb09, all the other data sets are the traces from P2P file sharing systems or OSN and are not published. So it is difficult to reproduce them. In the works using TREC, MEDLINE, Reuters or ClueWeb09, documents are distributed either by random or by the additional information like category or URL. Among them, works using MEDLINE, Reuters and TREC-9/6 distribute documents to peers according to their categories, which can make sure

[4] http://www.lemurproject.org/clueweb12.php/

[5] http://www.wikipedia.org/, available for download from http://download.wikimedia.org

[6] Medline database, US National Library of Medicine, http://www.ncbi.nlm.nih.gov/

[7] BM25 is a ranking function used by search engines to rank matching documents according to their relevance to a given search query

[8] https://delicious.com/

[9] http://www.citeulike.org/faq/data.adp

Table 5.1: Data sets used in state of the art (N/A: information is not available)

	Data set	Distribution strategy	Query	Relevant documents	No. of peers
[Tang 2003b]	TREC-7,8	Random	Predefined	Predefined	1000-10,000
[Balke 2005]	TREC-5	Random	Random words	N/A	100-2000
[Papapetrou 2007]	MEDLINE	Category	N/A	N/A	500-5000
[Papapetrou 2010]	Reuters	Category	N/A	N/A	1000-5000
[Podnar 2007, Skobeltsyn 2006]	Wikipedia	Random	Query log	Centralized BM25	4-28,1-1000
[Zhang 2007]	Gnutella trace	User	Query log	Term matching	1706
[Voulgaris 2007]	eDonkey trace	User	File name	exact searching	11,872
[Bender 2007, Bertier 2010]	OSN trace	User	Tags	Tag match	13,515/100,000
[Raftopoulou 2010]	TREC-9/6	Category	N/A	N/A	2000
[Doulkeridis 2010]	Reuters	N/A	Random words	Key word match	2000-20,000
[Ke 2010]	ClueWeb09	Web site(URL)	Documents	Exact searching	100-10,000

the documents in each peer have central topics. These central topics can be considered as the user's interests. In addition, most of the date sets are used to simulate the networks ranging from 100 to 10000. For simulating the networks with more than 10000 peers, Reuters and traces from eDonkey and OSN are used in [Voulgaris 2007, Bertier 2010, Doulkeridis 2010]. For query and relevant documents (ground truth), most of the works obtain the queries and relevant documents based on the data set they use and the searching task they target.

We choose Reuters for our simulation, considering that (i) Reuters is a freely available full-text data set, so that full-text P2P-IR can be performed and the simulation can be reproduced easily; (ii) Documents in Reuters have the additional information about their category, and the category information can be used for distributing documents to peers. (iii) Reuters has more than 800,000 text documents which can be distributed to a large number of peers, while the other two data sets MEDLINE and TREC-9/6, which also have category information for documents, have 348,566 and 30,000/556,078 documents, respectively.

5.1.3 Using Reuters Corpus

Reuters corpus is a collection of news articles in different categories. It consists of 804,414 news articles belonging to 103 categories (an article may belong to several categories). The 103 categories are hierarchical. By removing the categories on higher hierarchy, we get 77 categories. Since we assume each peer has one interest (as described in Section 3.1.1 in Chapter 3), we assign each peer a collection of documents D which are randomly extracted from one random category. For the value of $|D|$, we follow the strategy in [Papapetrou 2010] which also use Reuters and the category information to distribute documents, and assign each peer 20 documents from the same category. To avoid duplicated documents returned to the same query during IR, peers in the same category are not allowed to share common documents. This could in turn results in worse IR performance, since there is only one copy of the relevant document in the network. Therefore, it is a tradeoff between traffic cost and IR performance.

Although it has no available queries and relevance assessment, a set of artificial queries and relevant documents can be generated easily, as showed in the works like [Balke 2005, Podnar 2007, Papapetrou 2010]. A set of documents can be randomly selected as queries for searching other similar files, and the corresponding relevant documents of each query can be obtained by a centralized search engine. The results archived by the centralized search engine are used as the relevance assessment, because more advanced searching techniques can be used in a centralized search engine, so the results are more close to the ideal performance. Moreover, even with the same searching techniques, a centralized search engine can provide ideal ground truth to P2P-IR, to evaluate if the query routing in P2P-IR guarantees that the query can reach all the peers storing the ground truth relevant documents. In this thesis, we randomly chose 100 documents from Reuters corpus as representative possible queries in the network, considering that there are 77 categories in the network. For each query, we employ Lucene[10] to perform IR over the whole document collection. The top 100 documents returned by Lucene are used as the ground truth.

5.1.4 Generating Peer Profile

Peer profile is represented as a set of topics for describing the interest of the peer. We use Latent Dirichlet Allocation (LDA) to generate representative topics for each peer, as stated in Section 3.1.1 in Chapter 3.

LDA is trained on a collection of documents sampled from peers. We use Java-

[10] http://lucene.apache.org/core/

based package MALLET[11] to train LDA model and infer topics for new documents. Specifically, we use the same server that works as a bootstrap server of the P2P network to train LDA model, and share the trained model to all peers in the network. The peers use the model to infer the topics of their documents locally once they join the network. The inferred topics with a probabilistic weight above threshold 0.1 are regarded as the peer's topics. For the implementing details, we follow the work proposed in [Draidi 2011].

5.2 Experimental Setup

The simulation is performed in four parts. In the first part, we simulate the process of network topology evolution from a random P2P network to a SON where peers with similar content are clustered up. In the second part, IR over the generated P2P network topologies is simulated. An evolution of IR performance is expected during the evolution of the network topology. In the third part, we simulate the topology evolution with dynamic behaviors of the peers. Peer joining is the focus of this thesis. We simulate the behavior of peer joining at different times during the network evolution process. The new peers are initially connected to some random peers. They then use local search to build short-range links to similar peers. We are interested in how the status of the network topology and the local search strategy would affect the efficiency to build short-range links for the new peers that join the network at different times. In the last part, the SA-based local search is simulated with different configurations.

In the rest of this section, parameters used for the simulation are described, followed with a detailed simulation procedure.

5.2.1 Configuring the Network: Setting Parameters

The configuration for the network and peer rewiring must be set before the simulation. All the parameters and their baseline values are listed in Table 5.2.

The parameters are carefully configured, by considering their potential effect on the quality of peer clusters and IR results, as well as the required time and traffic cost. A detailed discursion based on experimental study will be presented in Section 5.4.5, a brief description about them is presented in the following:

Network size N refers to the scale of the network we study. In this thesis, we are interested in studying the evolution of overlay network with large number of

[11] http://mallet.cs.umass.edu/

Table 5.2: Parameters for the network configuration and SA-based local optimization

Networking	Network size	N	5000-25000
	Short-range links	s	15
	Long-range links	l	15
	Intra-distance threshold	θ	0.5
	TTL of clustering message	k_c	7
	TTL of query message	k_q	3,4
SA-based local optimization	Initial temperature	T_0	100
	Cooling schedule	$T(t)$	$T_0 a^t, 0 < a < 1$
	Relaxation time	k_r	7

peers. As the simulated networks in the state of the art often has 100-10000 peers, we simulate P2P networks with up to 25000 peers.

Number of long-range contacts l is to make the whole network connected. The number of long-range contacts affects the shortest path length between any two peers, and may hence affect the performance of peer clustering and IR. Experiments in Section 5.4.5 show that in order to make the network be a connected component, the minimum number of random links each peer should keeps is 11; in Table 5.14 in the same section, an explicit IR improvement can be observed when we change the value of l from 10 to 15. So l is set as 15 in the simulation, so that a good IR performance can be obtained while the maintenance cost of the links remains low. The value of l is the same for all the peers, since we assume peers have equal capability to maintain their links (considering maintaining a link requires periodical message exchanges).

Number of short-range contacts s is to make the peers with similar contents connected. It may affect the quality of peer clusters and the performance of intra-cluster IR. It is set as 15 after a tradeoff between the IR performance and the cost to maintain the links, as showed in Table 5.12 and Table 5.13 in Section 5.4.5. Similarly, the value of s is the same for each peer.

Intra-similarity threshold θ may affect the performance of Information Retrieval. It is set as 0.5, which achieves the best IR performance according to experimental study (part of the experimental results are reported in Table 5.12, Section 5.4.5).

TTL of clustering message k_c may affect the convergence time of peer clustering

and the quality of the resulting peer clusters.

TTL of query message k_q affects the performance of IR. To choose the value for these two TTL, a tradeoff has to be made between the traffic cost and the performance. According to experiments and a careful tradeoff, we set k_c as 7, and k_q as 3 and 4.

For SA-based local optimization, the following parameters are set:

Initial temperature T_0 is a simulated value to indicate the initial state of the system. We tried 50,100,500,1000 as an initial temperature, the overall performance with initial temperature 100 is better than the others with respect to their convergence speed in building SONs.

Cooling schedule $T(t)$ controls the cooling process of the temperature. It decides how the temperature decreases step by step. In Chapter 4, a specific analysis is made for selecting the appropriate cooling schedule, and the exponential cooling schedule is chosen.

Relaxation time k_r is set to be equal to k_c, as discussed in Section 4.5.3 of Chapter 4. It indicates the number of local search steps performed at a given temperature. This parameter may affect the quality of peer clusters.

5.2.2 Simulation Procedure

To start, we initiate a random P2P network with 25000 peers, each has $s + l$ links, as illustrated in Algorithm 11.

Then we execute peer rewiring repeatedly and independently in each peer. To facilitate observation, a predefined time interval τ is assigned as the period of rewiring cycles. During this period, each peer is randomly scheduled to initiate a peer rewiring cycle. After a period of τ, these peers can initiate another cycle of peer rewiring. Note that in each period, the order to initiate rewiring cycle might affect the rewiring performance of individually peer, but has slight effect on the overall clustering performance of the network thanks to the randomness. This will be discussed in detail in Section 5.4.5.

The baseline parameter sets are presented in Table 5.2. Table 5.3 summaries all the approaches we implement for peer rewiring. We implement the baseline local search approaches using random walk, greedy walk, and random/greedy walk each is employed with equal probability. These approaches are commonly used in the state of the art, such as in [Voulgaris 2007, Parreira 2007, Raftopoulou 2010]. SA-based

Algorithm 11: Generating random P2P network

```
1 Initiation: G = ⟨P, L⟩;
  // P refers to all the peers in the network, L is empty
2 for p_i ∈ P do
3     j = 0; // use j to count the number of the links p_i already has
      // p_i has to build s + l links, s short-range links and l
         long-range links
4     while j < s + l do
5         Randomly select a peer p_j from P − {p_i};
6         if ⟨p_i, p_j⟩ is not linked then
7             L = L ∪ {⟨p_i, p_j⟩};
8             j++;
```

approach is simulated with different value of a in the cooling schedule, and with both Metropolis and Glauber dynamics.

While the network topology evolves from a random network to a SON, IR is performed over a set of network topology sampled from the network evolution. In this thesis, the performance of IR is mainly to verify the quality of peer clusters, so only the IR performance within the clusters are considered. In each sampled network topology, 100 queries (100 random documents as described in Section 5.1.3) are initiated by the peers which have the queries' original copy in their document collections. Since the queries are randomly sampled, they can represent the possible queries in the simulated P2P network. Therefore, the P2P-IR result of these queries can exhibit the overall IR performance in the network. Since a query is also a part of its initiator's content, we consider the query, the query initiator and the peer cluster it belongs to share the similar topics. Therefore, the relevant documents to the query can be found in the peers that are in the same cluster with the initiator. To implement this, the query is diffused to the query initiator's neighborhood along the short-range links: it is forwarded to the peers that have a distance below or equal to θ to the query initiator (Algorithm 3 in Chapter 3). The query can be forwarded up to k_q hops maximally. For refining the performance, more complicated query routing strategy can be used within the peer cluster, such as the querying routing based on a routing table [Kumar 2005b] or social relations [Bender 2007]. However, this is not the focus of this thesis.

Besides simulating peer rewiring in a static network with a fixed number of peers, we also simulate the rewiring process in the network with dynamics. There are three types of dynamics: new peers joining the network, peers leaving the network and

Table 5.3: A summary of local search strategies

Baseline	Random walk
	Greedy walk
	Random/Greedy walk
SA-based local search	Metropolis dynamics
	Glauber dynamics

peers changing their content. For peers leaving the network and peers changing their content, the affected peers may simply rewire their links by explore similar peers via its current similar neighbors. But for new peers joining the network, they have to build connection to similar peers or peer clusters through their initially random links. In this thesis, we focus on simulating the rewiring process for new peers, because this is more challenging to find similar peers along random links than find similar peers via existing similar neighbors. In total, 365 new peers are simulated to join the network at a certain time during the network evolution. The contents of the new peers are duplicated from the contents of a set of existing peers, which are sampled uniquely from the 77 categories they belong to. The unique sampling makes the rewiring behaviors of these 365 new peers representative to the overall joining behaviors in the network. Different local search strategies are simulated to rewire the links of the new peers, including random walk, greedy walk, random/greedy walk and our SA-based local search.

In the end, we conduct the simulation of building SONs with various network configurations, in order to study how the network configuration affect the quality of the resulting SONs and the IR performance in SONs. We also perform the simulation in various random networks, in order to verify the robustness of our approach.

5.3 Metrics

We aim to evaluate the efficiency and effectiveness of the proposed peer clustering approach. The former involves in the time consumed for clustering peers, and the latter refers to the quality of peer clusters. The consumed time is evaluated as the number of rewiring cycles the peers take in order to achieve the short-range links meeting a certain criterion. For the quality of peer clusters, we employ three metrics: relative intr-cluster similarity, clustering efficiency and IR performance. As stated in Chapter 3, we define the task of peer clustering as an optimization problem with the objective function:

$$O(G) = \frac{1}{N}\sum_{i=1}^{N}(\overline{IntraDis}(p_i)) + \frac{1}{N}\sum_{i=1}^{N}(\overline{C}(p_i)), \tag{5.1}$$

It contains **relative intra-cluster similarity** of the network, which is the average value of each peer's relative intra-cluster similarity:

$$\overline{IntraDis}(p_i) = \begin{cases} 0, \text{ if } IntraDis(p_i) \leq \theta \\ \theta - IntraDis(p_i), \text{ if } IntraDis(p_i) > \theta \end{cases} \tag{5.2}$$

and **clustering efficiency** of the network, which is the average value of each peer's clustering efficiency:

$$\overline{C}(p_i) = \frac{|\{p_j \,|\, H(p_i, p_j) \leq \gamma, Dis(p_i, p_j) \leq \theta\}|}{|\{p_k \,|\, p_k \in G.P, Dis(p_i, p_k) \leq \theta\}|}. \tag{5.3}$$

Relative intra-cluster similarity and clustering efficiency are integrated in this objective function. They evaluate peer clusters from both local and global viewpoints, since the former only considers the peer and its short-range contacts, while the latter considers the peer and all the other similar peers in the network. In order to study the relationship between these two elements, we consider them as two different metrics.

To evaluate the quality of the peer clusters, we also evaluate the potential IR performance within the peer clusters. Since peers in a cluster have the content with similar topics, we expect that all the relevant documents for a query are located in the peer cluster. Therefore, for each query, we calculate the recall of the relevant documents, that is the number of the relevant documents that can be retrieved from the peer's neighborhood, divided by the total number of relevant documents in the network. The overall IR performance is measured as the average recall of all queries. Assuming q_i is a query initiated by p_i, D_{q_i} is its relevant documents located in the network, and p_k is any peer that receives the query q_i in the relevant peer cluster, the **recall** of q_i is calculated as:

$$R(q_i) = \frac{|\{d_j \,|\, d_j \in D_{q_i} \wedge d_j \text{ is stored in } p_k\}|}{|D_{q_i}|}. \tag{5.4}$$

5.4 Results and Analysis

5.4.1 Building SONs from Random Networks

In this section, we present the simulation results of building SONs from random networks, using the local search strategies showed in Table 5.3. We present and

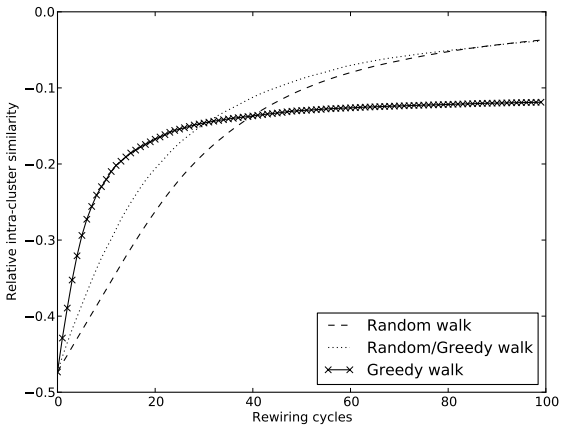


Figure 5.1: Optimization of relative intra-cluster similarity via random/greedy walk

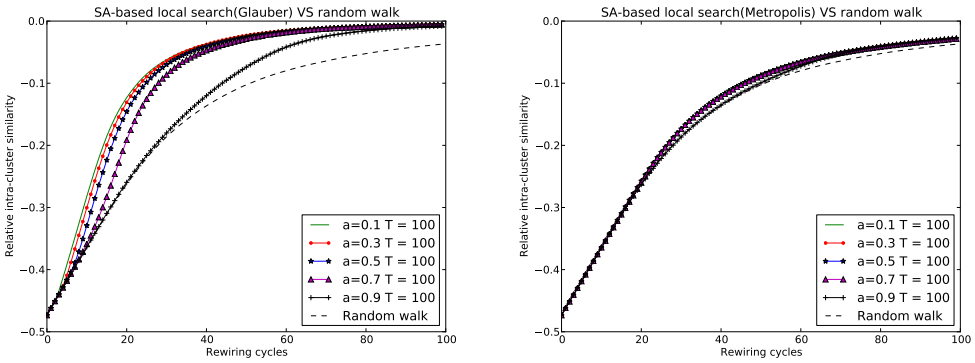
analyze the optimization process of the relative intra-cluster similarity and the clustering efficiency of the networks as the network topology evolves.

5.4.1.1 Relative Intra-cluster Similarity

Figure 5.1 shows the comparison of random walk, greedy walk and random/greedy walk with respect to the maximization process on the relative intra-cluster similarity of the network. Even though the greedy walk reaches a relatively high value very quickly, it ultimately performs the worst, because it is stuck in that status without getting much improvement ever since.

Comparatively, random walk and random/greedy walk perform better. They achieve almost the same relative intra-cluster similarity after 100 rewiring cycles, but the latter has a high optimization speed than the former in the beginning. As showed in the figure, up to 70 rewiring cycles, random walk keeps achieving a lower relative intra-cluster similarity than random/greedy walk. This is due to the feature of greedy walk, which can collect good peers efficiently when peer clusters emerge in the network. However, greedy walk alone is not able to optimize the relative intra-cluster similarity, because of its drawback of not being able to jump out of the local optima.

Figure 5.2 shows the optimization of relative intra-cluster similarity via SA-based local search with Glauber dynamics and Metropolis dynamics. The result of random walk is also presented for comparison. The following conclusions can be drawn from these two figures: (i) Using Glauber dynamics, SA-based local search can achieve a higher value of relative intra-cluster similarity, as well as higher convergence speed of optimization. In other words, SA-based local search with Glauber dynamics takes



(a) SA with Glauber dynamics (b) SA with Metropolis dynamics

Figure 5.2: Optimization of relative intra-cluster similarity with SA-based local search (random walk as the reference)

less rewiring cycles to achieve a certain value of relative intra-cluster similarity, comparing to the random walk and SA-based local search with Metropolis dynamics. (ii) Using Metropolis dynamics, SA-based local search performs slightly better than the random walk approach. (iii) In the results of SA-based local search with Glauber dynamics, with the same initial temperature $T_0 = 100$, small values of a can achieve higher convergence speed of optimization than large values.

More detailed comparing results are listed in Table 5.4 and Table 5.5, which describe the relative intra-cluster similarity of the network after different numbers of rewiring cycles. In Table 5.4, detailed results of SA-based local search with Glauber dynamics are presented. We can observe that SA-based local search with Glauber dynamics achieves the relative intra-cluster similarity -0.0051 after 100 rewiring cycles, while random walk achieves -0.03 after the same number of rewiring cycles. The standard deviation of the former is also smaller than that of random walk. Moreover, a high convergence speed can be observed with $a = 0.1$ and $a = 0.3$. For example, with $a = 0.1$, the relative intra-cluster similarity reaches -0.0364 with standard deviation 0.1038 after 40 rewiring cycles. The similar result is achieved by random walk after 100 rewiring cycles, by SA-based local search with $a = 0.5$ and $a = 0.7$, and by SA-based local search with $a = 0.9$ after 70 rewiring cycles.

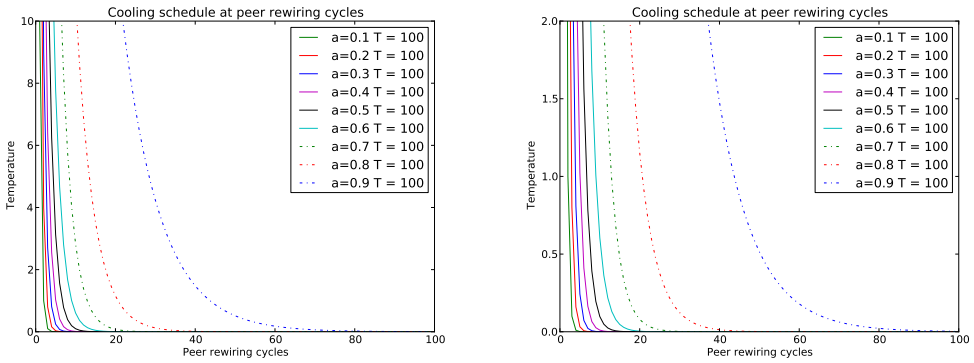
Table 5.5 presents the results of SA-based local search with Metropolis dynamics. No big difference can be observed between the approaches with different parameters. SA-based local search with Metropolis dynamics performs slightly better than random walk, and Metropolis dynamics with small a is slightly faster than that with large a. For example, when the rewiring cycles are between 20-70, the relative intra-cluster similarity with $a = 0.1$ is slightly higher than that with $a = 0.9$.

Table 5.4: SA-based local search with Glauber dynamics: relative intra-cluster similarity/standard deviation

	SA-based local search with Glauber dynamics, $T_0 = 100$					Random
	$a = 0.1$	$a = 0.3$	$a = 0.5$	$a = 0.7$	$a = 0.9$	walk
$RC = 0$	-0.4734 /0.0365	-0.4734 /0.0365	-0.4734 /0.0365	-0.4734 /0.0365	-0.4734 /0.0365	-0.4734 /0.0365
$RC = 10$	-0.2820 /0.1439	-0.3004 /0.1365	-0.3291 /0.1248	-0.3585 /0.1092	-0.3632 /0.1059	-0.3648 /0.1051
$RC = 20$	-0.1232 /0.1581	-0.1308 /0.1595	-0.1457 /0.1621	-0.1916 /0.1630	-0.2595 /0.1473	-0.2625 /0.1468
$RC = 30$	-0.0622 /0.1281	-0.0640 /0.1294	-0.0700 /0.1342	-0.0862 /0.1436	-0.1800 /0.1563	-0.1862 /0.1568
$RC = 40$	**-0.0364 /0.1028**	**-0.0376 /0.1037**	-0.0404 /0.1081	-0.0471 /0.1147	-0.1201 /0.1471	-0.1364 /0.1513
$RC = 50$	-0.0224 /0.0826	-0.0231 /0.0839	**-0.0255 /0.0880**	**-0.0286 /0.0922**	-0.0741 /0.1280	-0.1025 /0.1410
$RC = 60$	-0.0148 /0.0688	-0.0154 /0.0698	-0.0166 /0.0722	-0.0181 /0.0750	-0.0419 /0.1053	-0.0799 /0.1299
$RC = 70$	-0.0107 /0.0592	-0.0110 /0.0595	-0.0117 /0.0615	-0.0122 /0.0625	**-0.0243 /0.0845**	-0.0643 /0.1197
$RC = 80$	-0.0083 /0.0526	-0.0083 /0.0519	-0.0088 /0.0539	-0.0091 /0.0544	-0.0156 /0.0696	-0.0523 /0.1100
$RC = 90$	-0.0066 /0.0474	-0.0064 /0.0457	-0.0070 /0.0481	-0.0070 /0.0479	-0.0107 /0.0584	-0.0434 /0.1011
$RC = 100$	-0.0054 /0.0427	**-0.0051 /0.0407**	-0.0056 /0.0428	-0.0055 /0.0424	-0.0079 /0.0506	**-0.0366 /0.0935**

Table 5.5: SA-based local search with Metropolis dynamics: relative intra-cluster similarity/standard deviation

	SA-based local search, Metropolis dynamics, $T_0 = 100$					Random
	$a = 0.1$	$a = 0.3$	$a = 0.5$	$a = 0.7$	$a = 0.9$	walk
$RC = 0$	-0.4734 /0.0365	-0.4734 /0.0365	-0.4734 /0.0365	-0.4734 /0.0365	-0.4734 /0.0365	-0.4734 /0.0365
$RC = 10$	-0.3650 /0.1050	-0.3639 /0.1064	-0.3644 /0.1059	-0.3643 /0.1060	-0.3650 /0.1048	-0.3648 /0.1051
$RC = 20$	**-0.2584 /0.1526**	-0.2568 /0.1536	-0.2573 /0.1532	-0.2600 /0.1505	**-0.2631 /0.1464**	-0.2625 /0.1468
$RC = 30$	**-0.1736 /0.1654**	-0.1737 /0.1657	-0.1734 /0.1656	-0.1738 /0.1662	**-0.1866 /0.1563**	-0.1862 /0.1568
$RC = 40$	**-0.1195 /0.1566**	-0.1213 /0.1575	-0.1202 /0.1576	-0.1214 /0.1577	**-0.1351 /0.1510**	-0.1364 /0.1513
$RC = 50$	-0.0861 /0.1435	-0.0884 /0.1442	-0.0872 /0.1440	-0.0880 /0.1444	-0.0994 /0.1407	-0.1025 /0.1410
$RC = 60$	-0.0649 /0.1302	-0.0661 /0.1307	-0.0657 /0.1306	-0.0661 /0.1310	-0.0724 /0.1292	-0.0799 /0.1299
$RC = 70$	-0.0508 /0.1184	-0.0514 /0.1186	-0.0511 /0.1187	-0.0513 /0.1189	-0.0527 /0.1173	-0.0643 /0.1197
$RC = 80$	-0.0408 /0.1079	-0.0409 /0.1076	-0.0412 /0.1080	-0.0413 /0.1085	-0.0407 /0.1064	-0.0523 /0.1100
$RC = 90$	**-0.0333 /0.0983**	**-0.0333 /0.0982**	**-0.0335 /0.0987**	**-0.0339 /0.0994**	**-0.0325 /0.0965**	-0.0434 /0.1011
$RC = 100$	-0.0275 /0.0901	-0.0272 /0.0896	-0.0275 /0.0900	-0.0279 /0.0909	-0.0265 /0.0880	**-0.0366 /0.0935**

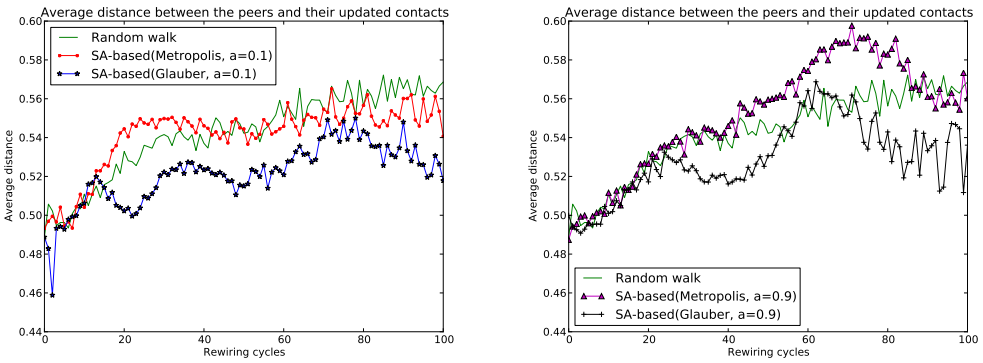


(a) Details for temperature 0-10 (b) Details for temperature 0-2

Figure 5.3: Cooling schedule with initial temperature 100 and different a

Now we explain (i) why small values of a perform better than high values of a and (ii) why the performances of Glauber dynamics is better than Metropolis dynamics. The different performance of different a is caused by the cooling schedule controlled by a. A cooling schedule with various a is showed in Figure 5.3. From this figure, it is clear that a cooling schedule with a large value of a decreases the temperature slowly. With $a = 0.9$, for example, we can observe that the temperature becomes below 2 after about 40 rewiring cycles. With $a = 0.7$, the temperature becomes below 2 after about 10 rewiring cycles. As we discussed in Chapter 4, a temperature higher than 2 results in almost a random walk for the local search. This can explain why the performance of SA-based local search is similar to that of random walk in the early phase of rewiring, specially for $a = 0.9$. After the temperature becomes below 2, more greedy walks and less random walks are taken. This trend continues as the temperature continues to decrease. This explicitly makes the performance with $a = 0.9$ start to be different from the random walk. Take an example in Table 5.4, from $RC = 40$, the performances of SA-based local search with $a = 0.9$ and random walk start to show explicit difference.

The different performance of Glauber dynamics and Metropolis dynamics can be explained by their different principles. In Glauber dynamics, both good and bad peers are accepted with some probability. In the beginning, good and bad peers are accepted with the similar probability. As the rewiring proceeds, less bad peers while more good peers are accepted. At the same temperature, the better the peer is, the higher the probability is to accept it. In Metropolis dynamics, instead, good peer is always accepted with probability 1, no matter how much improvement it



(a) a=0.1 (b) a=0.9

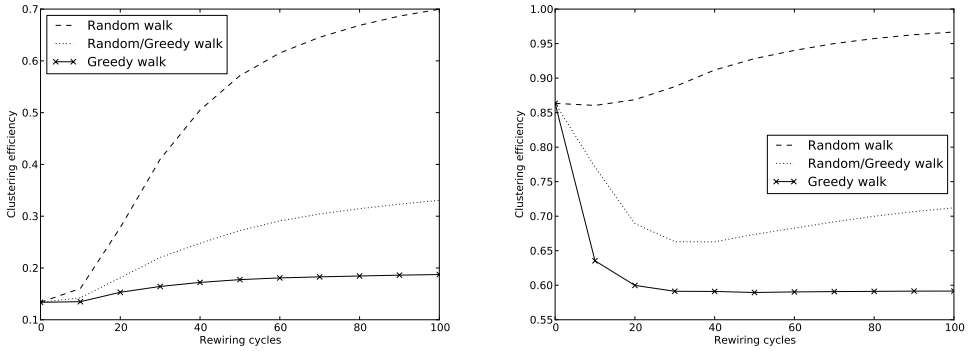
Figure 5.4: Analysis of peer rewiring behaviors: average distance between peers and their updated contacts

causes. Bad peers are accepted with high probability in the beginning and then are accepted with a gradually lowered probability. How these different principles make the rewiring performance different? We show an experimental analysis in Figure 5.4.

Figure 5.4 presents the peer rewiring behaviors of Glauber and Metropolis dynamics in the network: the average Jaccard distance between the peers and their updated contacts after each rewiring cycle. Only the behaviors with $a = 0.1$ and $a = 0.9$ are showed (Figure 5.4(a) and Figure 5.4(b), respectively), since these two values are typical among all the values. The similar pattern can be observed from these two figures: the average Jaccard distance in the algorithm with Glauber dynamics is generally smaller than that in the algorithm with Metropolis and random walk. It implies that the good peers accepted by Glauber dynamics have better quality than the good peers accepted by Metropolis dynamics, with respect to their distance to the rewiring peers. This allows SA-based local search with Glauber dynamics achieves the higher relative intra-cluster similarity using less rewiring cycles.

5.4.1.2 Clustering Efficiency

Figure 5.5 and 5.6 report the optimization process of clustering efficiency with different local search strategies. The clustering efficiency is calculated within the neighborhood with radius of $\gamma = 3$ and $\gamma = 4$ respectively. For the strategies of random walk, greedy walk and random/greedy walk, Figure 5.5 shows the compared results. Random walk performs the best. According to the clustering efficiency with $\gamma = 3$ in Table 5.6, random walk achieves the clustering efficiency 0.6997 on average after 100 rewiring cycles. It is much higher than that of greedy walk and random/greedy walk.



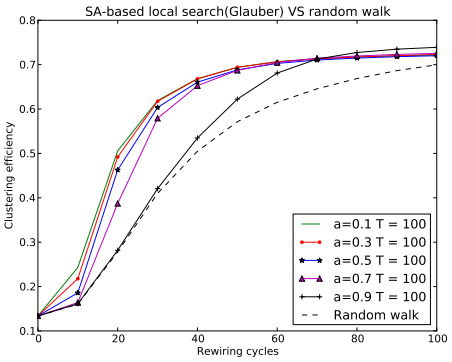
(a) $\gamma = 3$ (b) $\gamma = 4$

Figure 5.5: Optimization of clustering efficiency with random/greedy walk

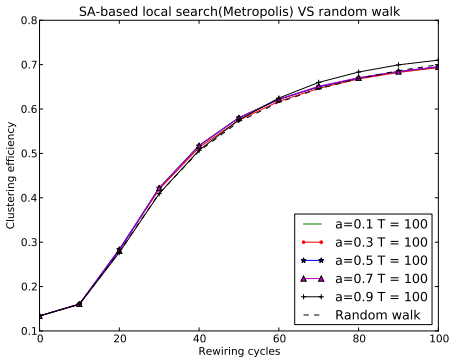
The reason of its high clustering efficiency mainly lies in its capability to find good peers by accessing bad peers. This capability is enabled by the fact that the walker for exploration the neighborhood can randomly steps to any possible peer, and can make connected the similar peers that are isolated by dissimilar peers. In this way, a higher clustering efficiency can be achieved because the similar peers can access each other within few hops via their short-range links. In addition, in Figure 5.5(b), the clustering efficiency decreases in the beginning, and increases afterwards. The same happens for the SA-based algorithm when $\gamma = 4$. The explanation will be given when we discuss the result of SA-based algorithms.

Comparing the results in 5.5 to the optimizing results of relative intra-cluster similarity in Figure 5.1, we can observe that, with the baseline approaches, a better performance in optimizing relative intra-cluster similarity does not imply a better performance in optimizing clustering efficiency. Specifically, random/greedy walk have a general better performance than random walk in optimizing the relative intra-cluster similarity; but in optimizing the clustering efficiency, random walk performs much better, because it better connects all the similar peers rather than only the peers and their short-range contacts.

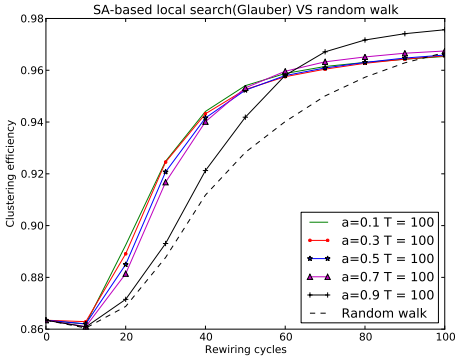
In Figure 5.6, the results of SA-based local search are demonstrated, with the results of random walk as the reference. For SA-based local search, the first thing we can observe is that Glauber dynamics outperforms random walk in terms of clustering efficiency and convergence speed. This observation is similar to our observation about their performance in optimizing the relative intra-cluster similarity, in which Glauber dynamics also outperforms random walk. Similar convergence pattern of clustering efficiency can also be observed from Table 5.7: SA-based local search with Glauber dynamics achieves the clustering efficiency 0.69 at 50 rewiring cycles



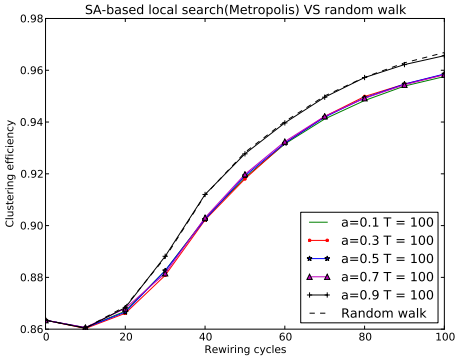
(a) SA with Glauber dynamics, $\gamma = 3$



(b) SA with Metropolis dynamics, $\gamma = 3$



(c) SA with Glauber dynamics, $\gamma = 4$



(d) SA with Metropolis dynamics, $\gamma = 4$

Figure 5.6: Clustering efficiency with SA-based local search (random walk as the reference)

Table 5.6: Clustering efficiency/standard deviation of random and greedy walk (RC: rewiring cycles, $\gamma = 3$)

	Local search approaches		
	Random walk	Random/greedy walk	Greedy walk
$RC = 0$	0.1338/0.0188	0.1338/0.0188	0.1338/0.0188
$RC = 10$	0.1604/0.0448	0.1424/0.0452	0.1349/0.0405
$RC = 20$	0.2781/0.1433	0.1811/0.0782	0.1531/0.0481
$RC = 30$	0.4100/0.1997	0.2202/0.0939	0.1643/0.0550
$RC = 40$	0.5049/0.2084	0.2477/0.0980	0.1721/0.0614
$RC = 50$	0.5714/0.2045	0.2722/0.0985	0.1775/0.0666
$RC = 60$	0.6152/0.1967	0.2909/0.0973	0.1808/0.0692
$RC = 70$	0.6456/0.1868	0.3044/0.0951	0.1828/0.0708
$RC = 80$	0.6686/0.1770	0.3145/0.0935	0.1845/0.0725
$RC = 90$	0.6865/0.1679	0.3233/0.0919	0.1861/0.0740
$RC = 100$	0.6997/0.1616	0.3307/0.0919	0.1873/0.0755

with a small standard deviation, while random walk achieves it after 100 rewiring cycles with a larger deviation. Similar to its performance in optimizing relative intra-cluster similarity, SA-based local search with Metropolis does not show much advancement. Its clustering efficiency with $\gamma = 4$ is showed in Table 5.8. From the table, we can see quite equal performances of SA-based local search with Metropolis and random walk. Since the results with $\gamma = 3$ and $\gamma = 4$ almost shows the same pattern, the results of Glauber dynamics with $\gamma = 4$ and the results of Metropolis dynamics with $\gamma = 3$ are not presented to avoid the redundance.

Furthermore, we can observe that small values of a perform better in general, but the highest clustering efficiency is always achieved by $a = 0.9$ after a certain number of rewiring cycles. For example, when $\gamma = 3$, with Glauber dynamics, $a = 0.9$ starts to achieve explicit better clustering efficiency after about 70 rewiring cycles; with Metropolis dynamics, $a = 0.9$ starts to outperform after about 55 rewiring cycles. When $\gamma = 4$, with Glauber dynamics, after about 70 rewiring cycles, $a = 0.7$ also performs better than the smaller values of a.

These improvements are probably because that $a = 0.9$ allows more random walks than the other values of a in the beginning of the rewiring process, which make similar peers be better connected. However according to the values showed in

Table 5.7: Clustering efficiency/standard deviation of SA-based local search with Glauber dynamics (RC: rewiring cycles, $\gamma = 3$)

	SA-based local search with Glauber dynamics, $T_0 = 100$					Random
	$a = 0.1$	$a = 0.3$	$a = 0.5$	$a = 0.7$	$a = 0.9$	walk
$RC = 0$	0.1338 /0.0188	0.1338 /0.0188	0.1338 /0.0188	0.1338 /0.0188	0.1338 /0.0188	0.1338 /0.0188
$RC = 10$	0.2431 /0.1193	0.2176 /0.0976	0.1858 /0.0692	0.1639 /0.0479	0.1611 /0.0447	0.1604 /0.0448
$RC = 20$	0.5056 /0.2158	0.4920 /0.2163	0.4628 /0.2143	0.3872 /0.1942	0.2818 /0.1432	0.2781 /0.1433
$RC = 30$	0.6192 /0.1921	0.6173 /0.1939	0.6035 /0.1978	0.5791 /0.2075	0.4206 /0.1992	0.4100 /0.1997
$RC = 40$	0.6684 /0.1645	0.6679 /0.1667	0.6606 /0.1696	0.6526 /0.1796	0.5348 /0.2039	0.5049 /0.2084
$RC = 50$	**0.6940 /0.1471**	**0.6940 /0.1493**	0.6881 /0.1507	0.6871 /0.1572	0.6223 /0.1926	0.5714 /0.2045
$RC = 60$	0.7068 /0.1373	0.7064 /0.1393	**0.7028 /0.1400**	**0.7047 /0.1431**	0.6811 /0.1721	0.6152 /0.1967
$RC = 70$	0.7135 /0.1315	0.7129 /0.1328	0.7104 /0.1335	0.7146 /0.1349	**0.7127 /0.1532**	0.6456 /0.1868
$RC = 80$	0.7172 /0.1276	0.7172 /0.1273	0.7148 /0.1289	0.7195 /0.1308	0.7274 /0.1437	0.6686 /0.1770
$RC = 90$	0.7198 /0.1244	0.7205 /0.1230	0.7179 /0.1246	0.7231 /0.1268	0.7349 /0.1374	0.6865 /0.1679
$RC = 100$	0.7219 /0.1219	0.7228 /0.1200	0.7201 /0.1216	0.7254 /0.1243	**0.7390 /0.1336**	**0.6997 /0.1616**

Table 5.8: Clustering efficiency/standard deviation of SA-based local search with Metropolis dynamics (RC: rewiring cycles $\gamma = 4$)

	SA-based local search, Metropolis dynamics, $T_0 = 100$					Random
	$a = 0.1$	$a = 0.3$	$a = 0.5$	$a = 0.7$	$a = 0.9$	walk
$RC = 0$	0.8635 /0.0218	0.8635 /0.0218	0.8635 /0.0218	0.8635 /0.0218	0.8635 /0.0218	0.8635 /0.0218
$RC = 10$	0.8606 /0.0467	0.8603 /0.0476	0.8605 /0.0484	0.8606 /0.0477	0.8604 /0.0471	0.8604 /0.0478
$RC = 20$	0.8670 /0.1078	0.8661 /0.1084	0.8666 /0.1088	0.8680 /0.1083	0.8683 /0.1079	0.8688 /0.1084
$RC = 30$	0.8826 /0.1409	0.8808 /0.1404	0.8826 /0.1400	0.8816 /0.1408	0.8881 /0.1371	0.8877 /0.1379
$RC = 40$	0.9027 /0.1491	0.9022 /0.1471	0.9024 /0.1486	0.9031 /0.1472	0.9120 /0.1419	0.9118 /0.1416
$RC = 50$	0.9185 /0.1433	0.9180 /0.1410	0.9193 /0.1414	0.9198 /0.1412	0.9277 /0.1364	0.9283 /0.1364
$RC = 60$	0.9315 /0.1318	0.9319 /0.1290	0.9318 /0.1303	0.9325 /0.1306	0.9397 /0.1259	0.9402 /0.1265
$RC = 70$	0.9412 /0.1205	0.9422 /0.1179	0.9419 /0.1184	0.9422 /0.1185	0.9496 /0.1144	0.9501 /0.1151
$RC = 80$	0.9483 /0.1100	0.9499 /0.1076	0.9493 /0.1082	0.9492 /0.1084	0.9572 /0.1036	0.9573 /0.1055
$RC = 90$	0.9539 /0.1008	0.9546 /0.1003	0.9547 /0.1001	0.9544 /0.1001	0.9622 /0.0956	0.9629 /0.0966
$RC = 100$	0.9576 /0.0943	0.9584 /0.0937	0.9586 /0.0934	0.9583 /0.0933	0.9656 /0.0891	0.9668 /0.0902

the tables, these improvements are so slight that they can be almost ignored.

In addition, the clustering efficiency decreases in the initial 10 rewiring cycles when $\gamma = 4$. This could be explained by the effect of clustering peers in a network. In our simulation, each peer has 15 connection as short-range links. These links are generated randomly in the beginning. Ideally, one peer may have contacts to $15^4 = 50625$ peers within 4 hops if no two peers can be reached repeatedly by taking the different paths. This number is even larger than the size of our network. Then, when similar peers are gradually connected, neighbors of a peer can become neighbors among themselves. Before the peer clusters are better formed, this can temporarily decrease the number of similar peers that can be accessed within 4 hops. This can be demonstrated by the analysis of the simulation results. Figure 5.7 illustrates the histogram of the clustering efficiency of all the peers and its evolution during the peer rewiring process. Since a lot of SA-based local search approaches perform similarly, only some are sampled and illustrated. From the figure we can observe: with $\gamma = 3$, the peaks of the histogram evolve from low clustering efficiency to high clustering efficiency; while with $\gamma = 4$, there are the peaks that move to the lower values of the clustering efficiency in the beginning of the evolution. It implies that in the beginning of the rewiring processes, a lot of peers can access less similar peers with their non-optimized short-range links, comparing to the number of the similar peers they can access with the random links before peer rewiring.

For SA-based local search with Glauber dynamics, the performance differences with various values of a can be explained by their cooling schedule. As we discussed in Section 5.4.1.1, when the temperature is above 2, the walking strategy of the rewiring message is close to a random walk, because only the very good (bad) peers are accepted (refused) with high (low) probability, the probability to accept the other peers are similar. A cooling schedule with different parameter is showed in Figure 5.3. From this figure, the cooling schedule with $a = 0.9$ decreases the temperature the most slowly. We can observe the temperature becomes below 2 after 40 rewiring cycles. This explains why its performance is similar to random walk in the beginning. After that, more greedy walks and less random walks are taken because of the decreasing temperature. This explicitly makes the performance with $a = 0.9$ different from the random walk. Moreover, the large number of random walks in the initial phase contributes a lot to avoid isolated peer cluster, which helps the approach achieves a better cluster efficiency in the end.

To summarize, our approach, SA-based local search with Glauber dynamics, shows its explicit advancement in optimizing clustering efficiency of the simulated SONs, thanks to its evolving local search strategy controlled by Glauber dynamics. This evolving local strategy allows a certain number of random walks to make con-

nected the similar peers in the network in the beginning, and then allows more greedy walks to efficiently find the optimal short-range contacts for each peer. Therefore, both random walk and greedy walk are properly employed during the evolution of the network topology, and then the performance of building SONs as well as the quality of the resulting SONs are improved.

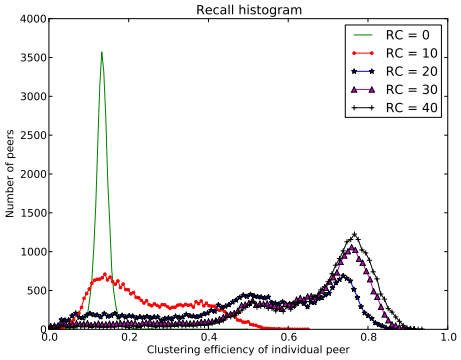
5.4.2 Information Retrieval in SONs

In order to verify the quality of the resulting SONs in the section above (Section 5.4.1), IR is performed within the generated peer clusters.

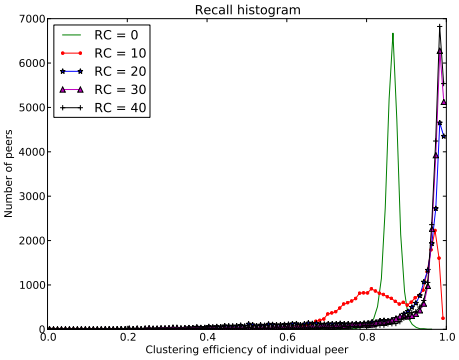
Since the query a peer initiates is from its own document collections (the configuration of IR are described previously in Section 5.2.2), the query is similar to the peer's documents. So it can be answered within the same peer cluster of the initiator. It is implemented by letting a peer to issue a query which is actually a document from its local content, and forwarding the query to the peers with a distance below θ along the initiator's short-range links. By doing this, the query is diffused in the peer cluster where the query initiator belongs to.

A maximum number of hops k_q is set for the query forwarding. In this simulation, we set is as 3 and 4. The performance with $k_q = 4$ is better than that with $k_q = 3$, but the performance difference between different approaches shows the similar pattern with $k_q = 3$ and $k_q = 4$. As stated in Section 5.3, we are mainly interested in how many relevant documents can be reached in the initiator's neighborhood, only IR recall is reported.

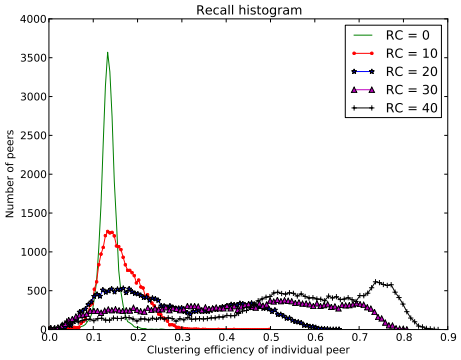
Figure 5.8 presents the IR recall with random walk, greedy walk and random/greedy walk. The best performance is achieved by random walk, as we expected, because it has a higher clustering efficiency and has good performance in avoiding isolated similar peer clusters. We can also observe a large standard deviation for the IR recall in Table 5.9. The similar observation can also be observed in the results achieved by performing IR in SONs generated by SA-based local search. This could be due to the heterogeneity of the queries we set up. The queries are the documents randomly extracted from the document collection of the whole network, where some queries have the relevant documents with high similarity, while others have the relevant documents with low similarity. At the same time, due to the non-uniform distribution of the documents with respect to their topics, in the network, we have peer clusters with different size. Some clusters may contain a large number of similar peers, while others are composed of only a few peers. This heterogeneity could make the location distribution of the relevant documents of the queries different from one another.



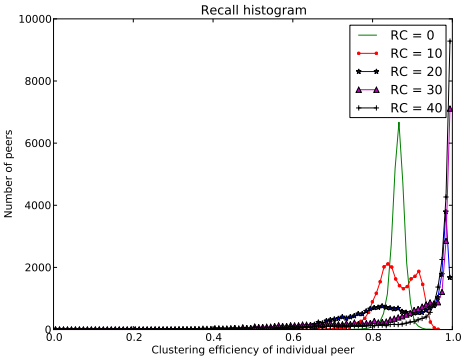
(a) Glauber,$a = 0.1$,$\gamma = 3$



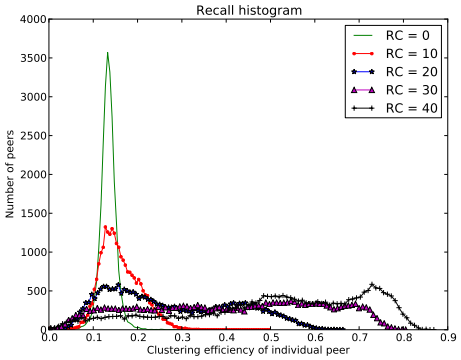
(b) Glauber,$a = 0.1$,$\gamma = 4$



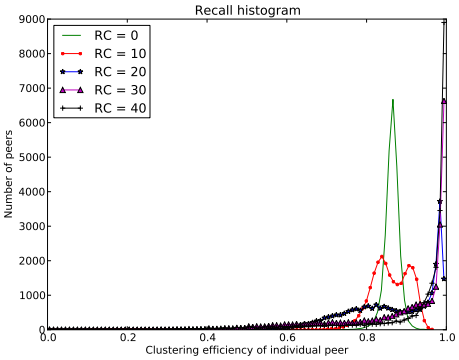
(c) Glauber,$a = 0.9$,$\gamma = 3$



(d) Glauber,$a = 0.9$,$\gamma = 4$

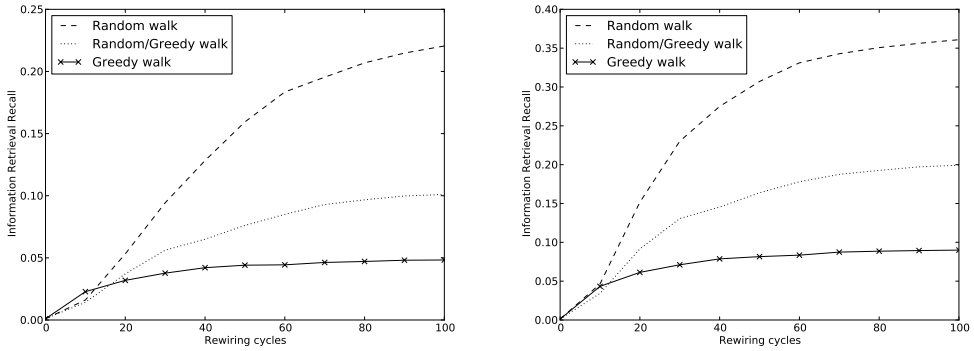


(e) Random walk,$\gamma = 3$



(f) Random walk,$\gamma = 4$

Figure 5.7: Evolution of the histogram of peers' clustering efficiency in different local search approaches (RC: rewiring cycles)



(a) $k_q = 3$ (b) $k_q = 4$

Figure 5.8: Evolution of IR recall in SONs generated via random/greedy walk

Table 5.9: IR recall/standard deviation in SONs generated with 100 rewiring cycles via random walk and greedy walk

	Local search approaches		
	Random walk	Random/greedy walk	Greedy walk
$k_q = 3$	0.2205 /0.1935	0.1010 /0.0951	0.0483 /0.0665
$k_q = 4$	0.3609 /0.3086	0.1993 /0.1900	0.0900 /0.1215

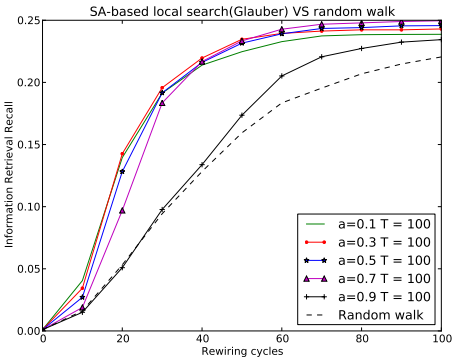
The IR recall keeps increasing as the network topology evolves from a random network to a SON, as we can observe from Figure 5.9. It implies that the network evolution makes it possible to access more peers storing the relevant documents. Moreover, the general IR performance with $k_q = 4$ is better than that with $k_q = 3$, since the similar peers accessed within 4 hops obviously are more than the peers accessed within 3 hops. This explanation can be confirmed by the results in Figure 5.6, which shows that the clustering efficiency with $\gamma = 3$ is about 0.72, while the clustering efficiency with $\gamma = 4$ is about 0.95.

However, the overall IR recall is low even when the final clustering efficiency with $\gamma = 4$ is about 0.95. This is the result of a tradeoff: the queries are only forwarded to the peers whose distance to the query initiator is below the threshold 0.5, in order to save the traffic cost. However, the relevant documents may be also stored in the peers whose distance to the query initiator is above 0.5. Moreover, in our simulation setting, two peers in the same category do not share any document, this could also be the reason of the low IR recall.

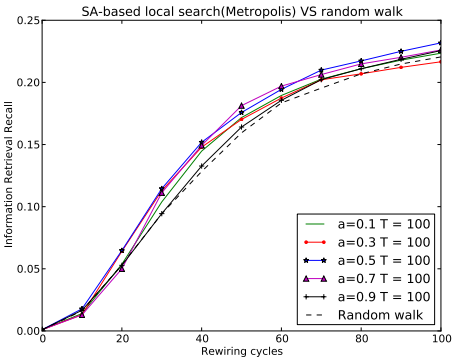
For SA-based local search, its IR recall is illustrated in Figure 5.9. The general IR recall of SA-based local search is higher than that of random walk. Among SA-based local search with different configurations, Glauber dynamics achieves good results with $a = 0.1, a = 0.3, a = 0.5$ and $a = 0.7$. SA-based local search with Metropolis does not show much IR improvement over random work, since it does not show much advancement over random walk in optimizing the clustering efficiency and the relative intra-cluster similarity.

More specific details about these results are presented in Table 5.10 and 5.11. Since the results with $k_q = 3$ and $k_q = 4$ show the similar pattern regarding the IR performance with different approaches, only the IR results with $k_q = 4$ are displayed in the table. We can observe from Table 5.10 that: to achieve the IR recall of 0.3609 with a deviation of 0.3086, random walk has to perform 100 rewiring cycles, while SA-based local search just needs 50 or 60 rewiring cycles. We can also observe the effect of many random walks in the beginning of SA-based local search with $a = 0.9$. It results in IR recalls similar to the results achieved by random walk. The data in Table 5.11 confirms our observation that SA-based local search with Metropolis dynamics is not much advanced than random walk.

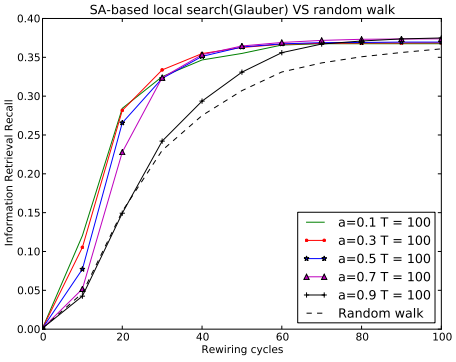
An overall low IR recall can also be observed from the results of SA-based algorithms, since the same tradeoff is used in order to save traffic cost: the queries are only forwarded to the peers whose distance to the query initiator is below the threshold 0.5. The other peers that are similar to the query initiator are not accessed even though they have the relevant documents. In general, from these results, we can conclude that network topology with similar peers clustered (SON) can im-



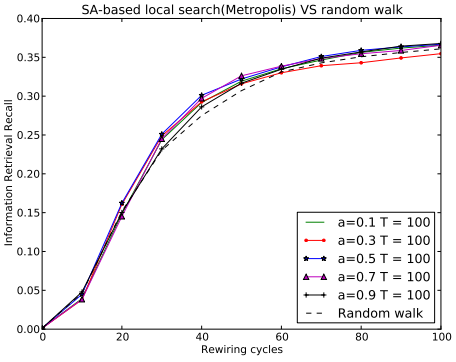
(a) SA with Glauber dynamics, $k_q = 3$



(b) SA with Metropolis dynamics, $k_q = 3$



(c) SA with Glauber dynamics, $k_q = 4$



(d) SA with Metropolis dynamics, $k_q = 4$

Figure 5.9: Evolution of IR recall in SONs generated via SA-based local search (random walk as the reference)

prove the subsequent IR performance. We can also conclude that the higher the clustering efficiency is, the better the IR performance it. However, a high clustering efficiency has limited effect on the IR performance if the querying routing technique is not properly designed. In order to have really good IR performance within a peer cluster, more advanced query routing techniques should be considered, such as routing table [Kumar 2005b] or additional relations among peers like social relations [Bender 2007].

5.4.3 Enhanced SA-based Local Search

In this section, we present the simulation results of the enhanced SA-based local search: its performance in building SONs and the IR recall in the resulting SONs. Enhanced SA-based Local Search aims to improve the performance of SA-based Local Search, by allowing the walkers to collect the information of more peers in a single local search process (the algorithm is presented in Chapter 4). SA-based Local Search with Glauber dynamics and with $a = 0.7$ is chosen as the baseline, since it shows a good performance in building SONs and results in a relatively high IR recall after 100 rewiring cycles. We implement enhanced SA-based local search using the same configuration of the baseline SA-based Local Search algorithm.

In Figure 5.10 and 5.11, compared results are presented for the SA-based local search and enhanced SA-based local search. We can observe that enhanced SA-based local search greatly improves the optimization process of relative intra-cluster similarity. Within less than 10 rewiring cycles, it achieves a quite high relative intra-cluster similarity which is very close to the optimum. As we stated before, in the very beginning of peer rewiring, peers are explored with a local search strategy similar to random walk. With random walk in a random network, an extensive search space is explored. Statistically, a large search space provides more good peers. Consequently enhanced SA-based local search is able to collect all the good peers in the extensive search space. This enables enhanced SA-based local search to find the appropriate short-range contacts quickly.

However, this good result has a cost: the quick optimization drives the network topology into a local minimum quickly, and makes it not be able to jump out from it. More specifically, enhanced SA-based local search tends to take all the similar neighbors of its neighbors as its own neighbors. This operation can easily results in the following connection configuration: peer p_a has short-range contact p_b, p_b's short-range contact p_c is also p_a's short-range contacts. If too many of this type of links appear in the network, peers can have the difficulty to explore the other part of the network and access the other similar peers in the network. This can be observed in Figure 5.11, which presents the optimization process of clustering

Table 5.10: IR recall/standard deviation in SONs generated with 100 rewiring cycles via SA-based local search with Glauber dynamics ($k_q = 4$)

	SA-based local search with Glauber dynamics, $T_0 = 100$					Random
	$a = 0.1$	$a = 0.3$	$a = 0.5$	$a = 0.7$	$a = 0.9$	walk
$RC = 0$	0.0012 /0.0045	0.0012 /0.0045	0.0012 /0.0045	0.0012 /0.0045	0.0012 /0.0045	0.0012 /0.0045
$RC = 10$	0.1204 /0.2054	0.1053 /0.1934	0.0772 /0.1570	0.0518 /0.1138	0.0425 /0.1008	0.0464 /0.1172
$RC = 20$	0.2843 /0.3026	0.2816 /0.3020	0.2656 /0.3002	0.2279 /0.2936	0.1493 /0.2398	0.1517 /0.2461
$RC = 30$	0.3251 /0.3043	0.3338 /0.3091	0.3228 /0.3117	0.3238 /0.3099	0.2421 /0.2991	0.2302 /0.2952
$RC = 40$	0.3466 /0.2996	0.3548 /0.3039	0.3512 /0.3027	0.3534 /0.3059	0.2936 /0.3083	0.2750 /0.3127
$RC = 50$	**0.3550 /0.3004**	**0.3627 /0.3039**	**0.3633 /0.3051**	**0.3646 /0.3059**	0.3309 /0.3118	0.3071 /0.3121
$RC = 60$	**0.3658 /0.3005**	0.3668 /0.3048	0.3673 /0.3069	0.3691 /0.3055	**0.3560 /0.3078**	0.3311 /0.3094
$RC = 70$	0.3675 /0.3019	0.3680 /0.3049	0.3689 /0.3073	0.3718 /0.3056	**0.3671 /0.3063**	0.3428 /0.3087
$RC = 80$	0.3675 /0.3021	0.3684 /0.3050	0.3691 /0.3076	0.3732 /0.3057	0.3710 /0.3069	0.3507 /0.3106
$RC = 90$	0.3675 /0.3021	0.3684 /0.3052	0.3696 /0.3077	0.3736 /0.3059	0.3737 /0.3076	0.3562 /0.3093
$RC = 100$	0.3675 /0.3021	0.3686 /0.3053	0.3696 /0.3077	0.3738 /0.3061	**0.3749 /0.3083**	**0.3609 /0.3086**

Table 5.11: IR recall/standard deviation in SONs generated with 100 rewiring cycles via SA-based local search with Metropolis dynamics ($k_q = 4$)

	SA-based local search with Metropolis dynamics, $T_0 = 100$					Random
	$a = 0.1$	$a = 0.3$	$a = 0.5$	$a = 0.7$	$a = 0.9$	walk
$RC = 0$	0.0012 /0.0045	0.0012 /0.0045	0.0012 /0.0045	0.0012 /0.0045	0.0012 /0.0045	0.0012 /0.0045
$RC = 10$	0.0393 /0.0860	0.0380 /0.0902	0.0449 /0.1162	0.0384 /0.0843	0.0479 /0.1098	0.0464 /0.1172
$RC = 20$	0.1484 /0.2364	0.1607 /0.2456	0.1624 /0.2503	0.1452 /0.2374	0.1502 /0.2399	0.1517 /0.2461
$RC = 30$	0.2439 /0.2808	0.2488 /0.2943	0.2510 /0.2924	0.2450 /0.2807	0.2321 /0.2898	0.2302 /0.2952
$RC = 40$	0.2918 /0.2970	0.2927 /0.3040	0.3011 /0.3086	0.2972 /0.3015	0.2862 /0.3030	0.2750 /0.3127
$RC = 50$	0.3193 /0.3028	0.3159 /0.3047	0.3221 /0.3121	0.3262 /0.2993	0.3167 /0.3076	0.3071 /0.3121
$RC = 60$	0.3352 /0.3050	0.3301 /0.3034	0.3377 /0.3096	0.3386 /0.3015	0.3346 /0.3076	0.3311 /0.3094
$RC = 70$	0.3465 /0.3017	0.3392 /0.3021	0.3510 /0.3087	0.3485 /0.3005	0.3489 /0.3063	0.3428 /0.3087
$RC = 80$	0.3563 /0.3009	0.3430 /0.3015	0.3590 /0.3071	0.3546 /0.2996	0.3569 /0.3056	0.3507 /0.3106
$RC = 90$	0.3619 /0.3027	0.3492 /0.3001	0.3634 /0.3072	0.3588 /0.3002	0.3644 /0.3039	0.3562 /0.3093
$RC = 100$	0.3656 /0.3028	0.3547 /0.2988	0.3669 /0.3086	0.3655 /0.3019	**0.3678 /0.3049**	**0.3609 /0.3086**

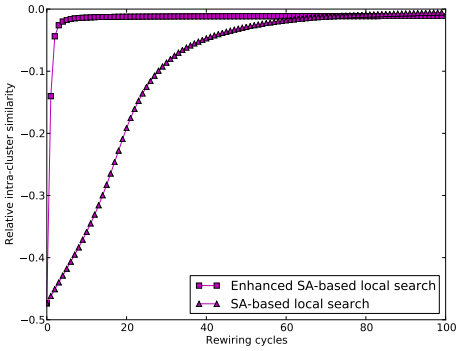
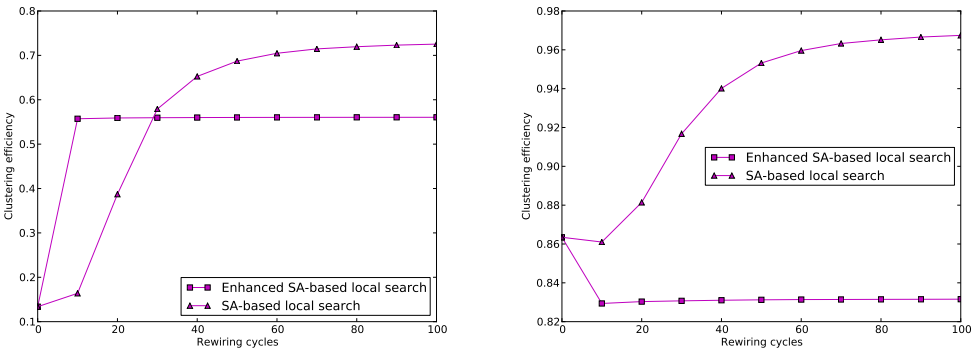


Figure 5.10: Comparison of enhanced SA-based local search and SA-based local search in relative intra-cluster similarity

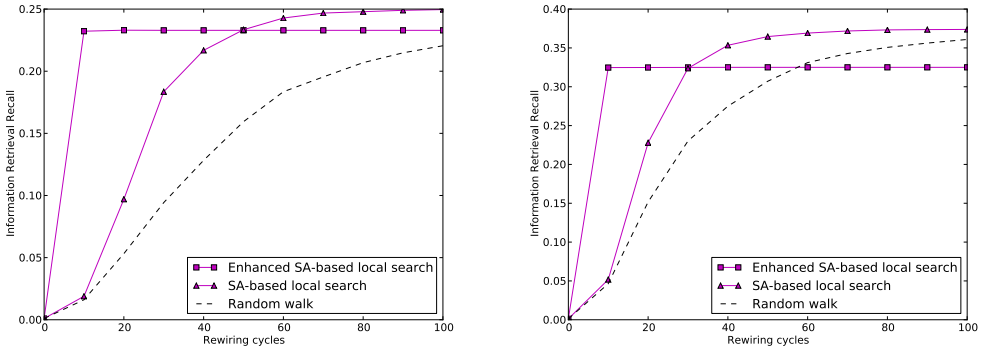


(a) $\gamma = 3$ (b) $\gamma = 4$

Figure 5.11: Comparison of enhanced SA-based local search and SA-based local search regarding clustering efficiency

efficiency achieved by enhanced SA-based local search and SA-based local search. With $\gamma = 4$, the enhanced SA-based local search does not improve the clustering efficiency. Instead it decreases the clustering efficiency and keeps it stable since then. With $\gamma = 3$, enhanced SA-based local search reaches a quite high clustering efficiency in a short time, but stops to make explicit improvement since then. Besides of the reason of the shrimped exploring space, this is also because as most of the peers achieve the required short-range links in the beginning, less peers start rewiring process in the network.

Regarding the IR recall of these two approaches (Figure 5.12), we can observe that enhanced SA-based local search can achieve an acceptable IR performance, but it is not able to break its maximum performance to make further improvement.



(a) $k_q = 3$ (b) $k_q = 4$

Figure 5.12: Comparison of enhanced SA-based local search and SA-based local search in IR recall

5.4.4 Building SONs with Dynamic Behaviors: Peer Joining

New peers are assumed to join the network during the evolution of the network towards SONs. In this experiment, the network evolution is implemented via SA-based local search with Glauber dynamics and $a = 0.1, T_0 = 100$, since it achieved good performance in building a SON. We conduct the new peers to join the network after 10, 20, 30, ... 80 rewiring cycles. The setup of the new peers was presented in Section 5.2.2. The rewiring behaviors of the new peers are then recoded and displayed in Figure 5.13 and 5.14. The following facts can be observed:

(i) the best optimization performance is showed in SA-based local search, particularly when the number of rewiring cycles reach 100. In addition, when the new peers join the network at or before 50 rewiring cycles, SA-based local search achieves almost the same relative intra-cluster similarity and clustering efficiency. When they join the network after 60 rewiring cycles, the optimization result at rewiring cycle 100 degrades. This is due to two reasons: less rewiring cycles are performed; the temperature of the system becomes so low that good peers are explored with probability of 1, while no bad peers are explored.

(ii) greedy walk tends to outperform the other strategies in optimizing the relative intra-cluster similarity in the initial phase of peer rewiring, and then is surpassed by the other approaches. This is because greedy walk tends to get stuck in local optimum and not be able to jump out of it. This conclusion is similar to the one we summarized by simulating the network evolution from a random network to a SON (Figure 5.1). From the simulation results in Figure 5.5, we also observed greedy walk performs the worst in optimizing the clustering efficiency. However, Figure 5.14 shows that greedy walk achieves comparable clustering efficiency for

the new peers, with respect to random walk and random/greedy walk. This results from the continuously changing situation of the network topology while the new peers rewire their links. As we described in the beginning of this section, as the new peers join the network and rewire their links, the peers in the network are becoming well clustered, due to our well-performed SA-based local search. The gradually clustered peers make it possible that greedy walk can keep finding good peers from the neighborhood. Oppositely, in the simulation of the network evolution from a random network to a SON, all the peers use greedy walk to rewire their links, so the network topology does not evolve well, and peers are not clustered well. Greedy walk in this case achieves bad results without any doubt.

(iii) random walk shows no advantage comparing with other approaches. Specially, when the new peers join the network after 60 rewiring cycles, random walk performs the worst. This is partially because greedy walk performs better for the new peers. In addition, it is also because random walk does not play the similar role in random networks and SONs. In a random network, random walk has the potential to explore more good peers, while in a SON, its potential is degraded due to the emergence of peer clusters.

(iv) but still, random walk plays an important role in making similar peers accessible to each other. That explains the good performance of random/greedy walk.

Similar conclusions can be drawn for the performance of clustering efficiency and IR recall within 4 hops, which is not presented for avoiding repetition.

5.4.5 Configuration VS. Performance

5.4.5.1 Link Number/TTL VS. Performance

In this subsection, we present how the number of links and the TTL of the walkers and queries affect the performance of peer clustering and the subsequent IR. Since the performance of these two tasks are correlated in the way that high clustering efficiency indicate better IR performance (Figure 5.6 and Figure 5.9), only the effect of the parameters on the performance of IR is reported, in order to avoid redundance.

IR in this subsection is performed by flooding the queries to all the peers within k_q hops along the short-range links, since short-range links are supposed to point to similar peers. We made this choice because this can exhibit the potential of a peer's neighborhood to answer its query which shares the similar topics with the peers in the neighborhood. In other words, a high IR recall by flooding can guarantee that the peers with the relevant documents is within the neighborhood. With this

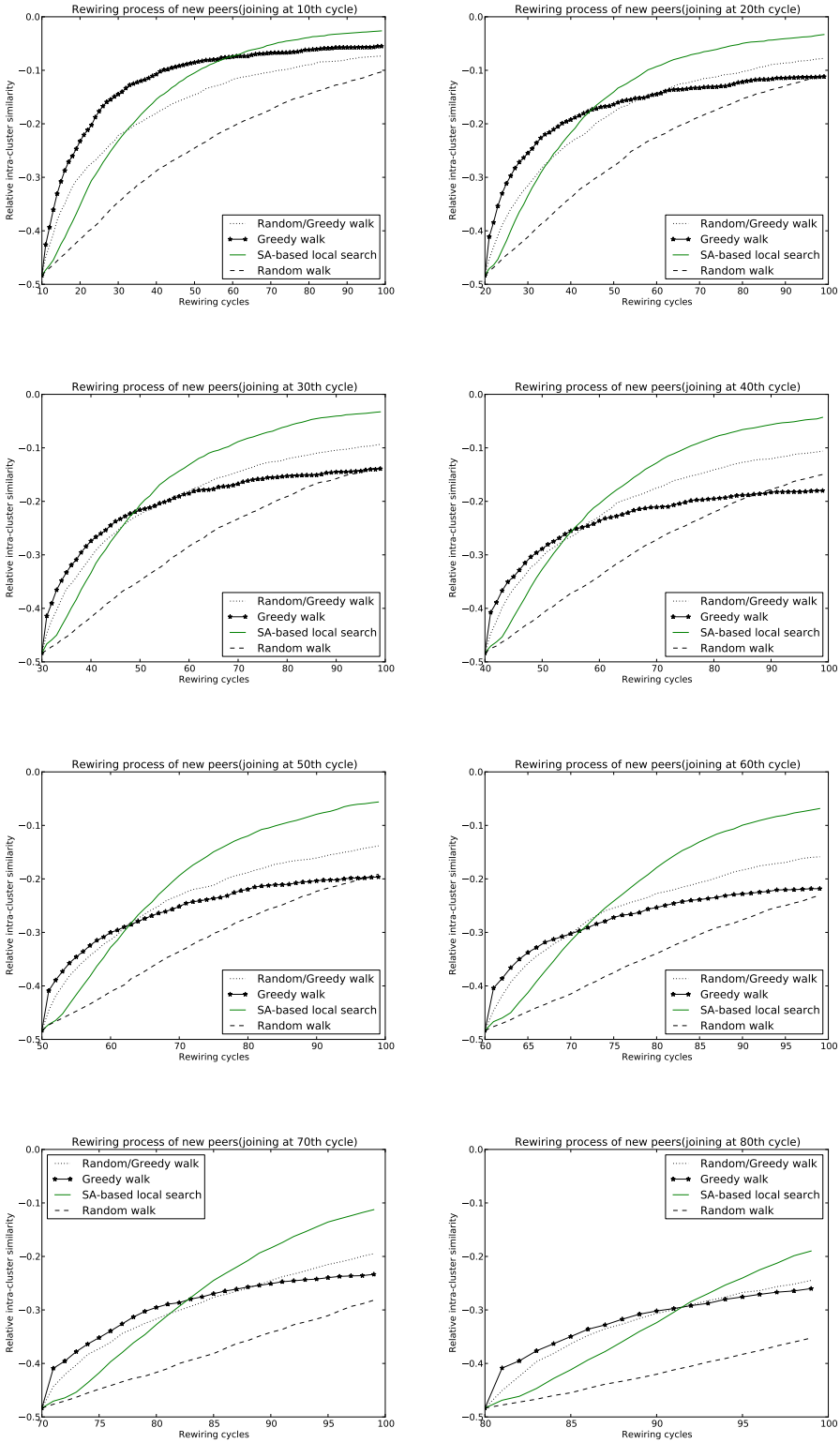


Figure 5.13: Relative intra-cluster similarity of new peers (RC: rewiring cycles)

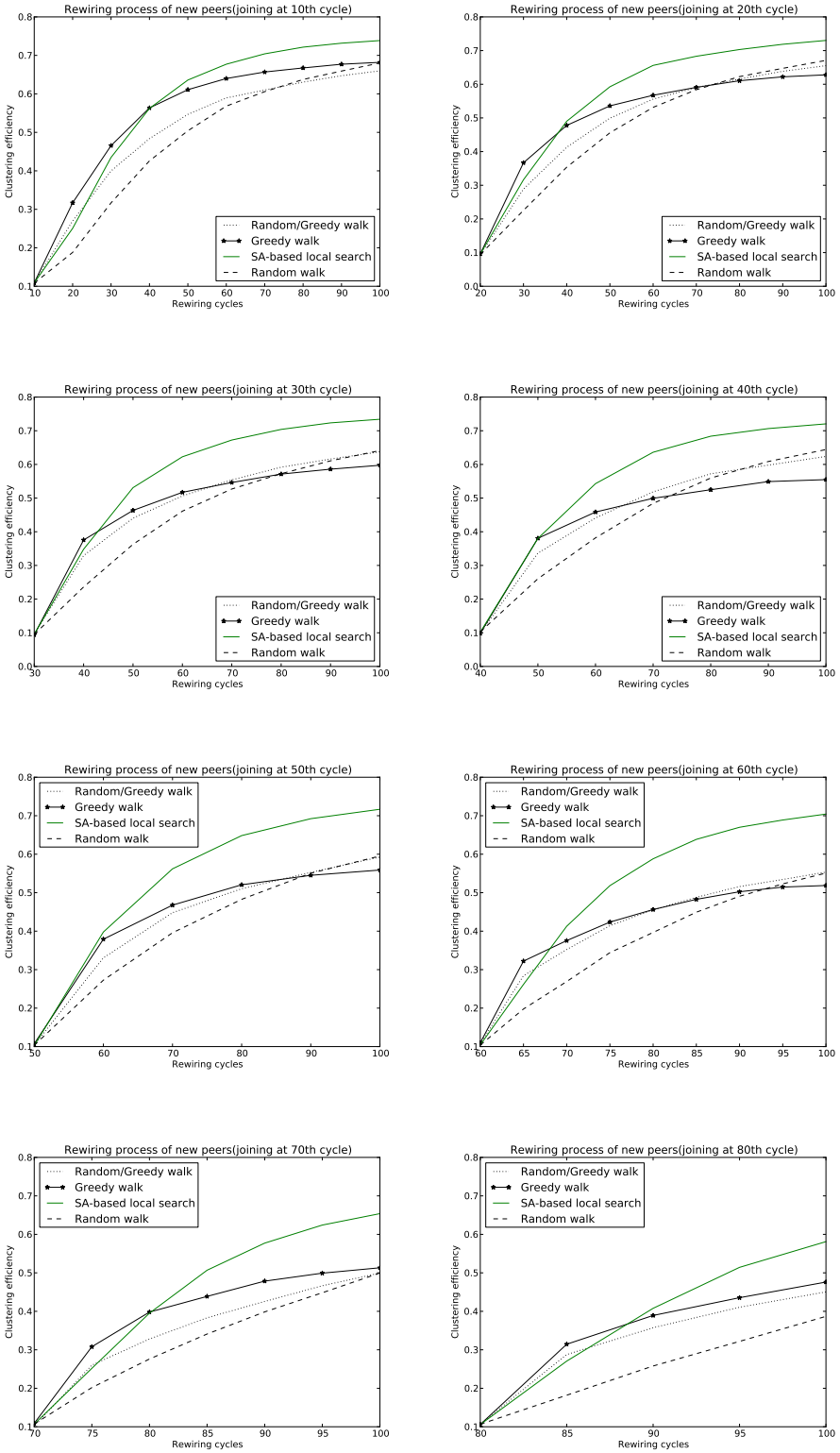


Figure 5.14: Clustering efficiency of new peers (RC: rewiring cycles, $\gamma = 3$)

Table 5.12: IR recall of s and θ with $N = 25000, l = 15, k_c = 7, k_q = 3$

		s				
		5	10	15	20	25
θ	0.2	0.031	0.065	0.089	0.106	0.121
	0.3	0.028	0.053	0.080	0.098	0.117
	0.4	0.032	0.087	0.140	0.176	0.219
	0.5	**0.051**	**0.163**	**0.257**	**0.336**	**0.424**
	0.6	0.035	0.096	0.168	0.246	0.306
	0.7	0.039	0.112	0.201	0.292	0.387

guarantee, advanced query routing approach can be designed to only forward the query to the peers with relevant documents within k_q hops, while reduce the traffic cost caused by flooding.

We firstly check the performance with different values of number of short-range contacts s, intra-similarity threshold θ, and TTL of query message k_q, by fixing the other parameters: the network size N, the number of long-range links l, the TTL of the walkers k_c and the maximum hops the queries are forwarded. These constant parameters make sure that the results with different s, θ and k_q are comparable. IR performance after 100 rewiring cycles is reported. The rewiring is performed using random walk, considering random walk is the best baseline approach and its performance over different parameters can be representative.

Table 5.12 shows the results with different combination of s and θ. We can observe the best performance is achieved by $\theta = 0.5$. This observation confirms that it is necessary to define a distance threshold between each peer and its short-range contacts, because a proper threshold can control the quality of the similar peers a peer can access via the short-range links. This is very important for IR. In addition, the results indicate that the larger the value of s is, the better the performance is. This is self-explaining, considering the fact that the more connections a peer has, the more peers it can access. So the more short-range contacts a peer has, the more similar peers it can access within k_q hops, and consequently the more relevant documents can be found.

With parameter θ fixed as 0.5, the performance of different values of s and k_q is reported in Table 5.13. The conclusion is: the higher the values of s and k_q are, the better the IR recall is. The explanation is quite obvious: with high values of s and k_q, more relevant peers are contacted, and surely the IR recall is better. In addition, we can also observe that when the value of s is small, increasing the value of k_q does not improve the IR recall as much as when the value of s is large. Similarly, when

Table 5.13: IR recall of s and k_q with $N = 25000, l = 15, k_c = 7, \theta = 0.5$

		s				
		5	10	15	20	25
k_q	2	0.023	0.053	0.072	0.103	0.133
	3	**0.051**	**0.163**	**0.257**	**0.336**	**0.424**
	4	0.107	0.320	0.490	0.560	0.658
	5	0.197	0.477	0.657	0.738	0.843

Table 5.14: IR recall of l and k_c with $N = 25000, s = 15, k_q = 4, \theta = 0.5$

		l				
		5	10	15	20	25
k_c	3	0.261	0.308	0.349	0.385	0.402
	4	0.307	0.351	0.392	0.409	0.437
	5	0.338	0.377	0.421	0.449	0.463
	6	0.384	0.421	0.446	0.458	0.489
	7	0.413	0.439	**0.490**	0.500	0.510
	8	0.445	0.475	0.506	0.517	0.530

the value of k_q is small, increasing the value of s also does not improve the IR recall as much as when he value of k_q is large. Therefore, a proper combination of s and k_q must be chosen in order to obtain acceptable IR performance.

Regarding the number of long-range contacts l and the TTL of clustering message k_c, different combinations of their values are studied with other parameters fixed as $N = 25000, s = 15, k_q = 4, \theta = 0.5$, since these parameters can generate quite good IR performance. From the result in Table 5.14, similar conclusion can be drawn: the larger l and k_c are, the better the IR performance is. With larger l and k_c, larger number of similar peers are clustered in the way that they can access each other within k_q hops. However, the IR improvement cause by larger l and k_c is not as explicit as in Table 5.13, especially when $l > 15$ and $k_c > 7$. Since the long-range links are used to keep peer cluster as well as the whole network connected, this observation may implies that the IR performance would not be greatly improved by the increasing number of the long-range links once the TTL of the rewiring message can allow one peer to access the other peers in the network.

According to the above evaluation with different parameters, only the threshold for intra-cluster distance can be decided by experimental results, because there exists

Table 5.15: Number of strongly connected components with different values of N and l

$N\backslash l$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
25000	24907	5008	1483	475	181	51	21	10	4	2	1	1	1	1
20000	19785	3971	1155	403	139	46	20	6	4	1	1	1	1	1
15000	14697	3020	898	293	124	32	12	4	1	2	1	1	1	1
10000	9939	2047	607	203	77	21	8	3	3	2	1	1	1	1
5000	4977	1024	288	107	27	14	3	1	4	1	1	1	1	1
1000	962	196	55	19	7	4	1	2	2	1	1	1	1	1
100	90	28	8	7	1	1	1	1	1	1	1	1	1	1
10	8	2	1	1	1	1	1	1	1	1	1	1	1	1

a value which achieves the best performance when the other parameters about the network are fixed. For the other parameters, the performance always gets better when the value of the parameter increases.

Since large parameters results in more maintenance cost and traffic cost, like large l and k_c, the appropriate parameters should be decided with a tradeoff between the cost and the IR performance. For example, for the number of long-range links, we expect a minimum number which makes all the peers are connected, formally called a strongly connected component [Dorogovtsev 2001]. This provides the possibility to cluster all the similar peers using peer rewiring, since a peer can access all the other peers. At the same time, a small number of long-range links requires a low maintenance load. In Table 5.15, we present the required number of links for each peer in order to have a strongly connected component. To implement this, we randomly build P2P networks with N peers and l connections for each peer using Algorithm 11. We then use the open source library iGraph [12] to check the number of strongly connected components in the generated random network. We can observe that a strongly connected component can be generated with $l > 10$ for each peer.

5.4.5.2 Network Topology/Size VS. Performance

In this subsection, a set of simulations is made to study the effect of network topology and network size on the performance of building SONs. Since SA-based local search with Glauber dynamics showed a good performance in building SONs and the subsequent IR, we also use it to build SONs in 4 random networks with 25000 peers (peers are the same for all the networks) and 30 random connections $(s+l)$ for each peer. The configuration of the other parameters follows the configuration in

[12] http://igraph.sourceforge.net/

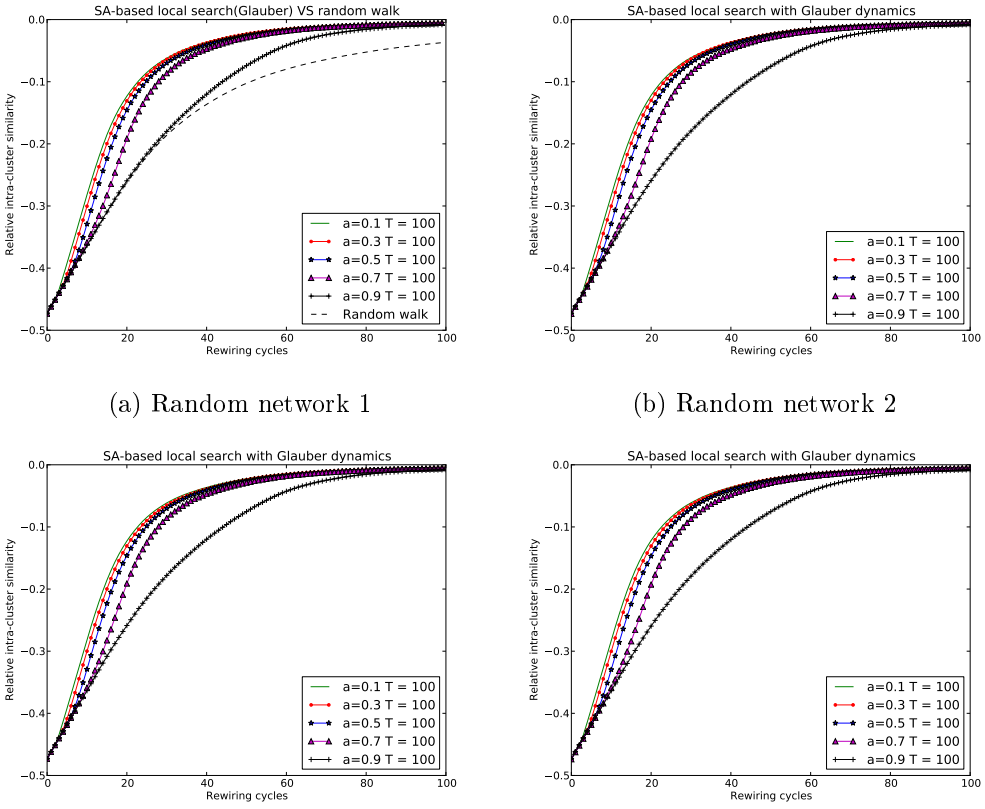
Table 5.2. We execute the same SA-base local search in another 2 random networks, with 10000 and 5000 peers, respectively, and compare their results with the results we achieved in the network with 25000 peers. The networks with 5000 and 10000 peers are generated with the same number of random connections for each peer. In order to keep a similar distribution of the peer clusters, these 5000 and 10000 peers are peers that are randomly sampled from the network with 25000 peers. For each peer in these three networks, there are on average 131, 262 and 656 peers whose distance to this peer is below the threshold $\theta = 0.5$. The performance of building SONs is exhibited by the relative intra-cluster similarity of the network, and the clustering efficiency of the network with $\gamma = 4$.

In Figure 5.15 and 5.16, we show the results of SA-based local search with Glauber dynamics in four initially different random networks with 25,000 peers. No significant difference can be observed from these results. Therefore, we can verify that the performance of our approach is robust to the randomness of the initial networks.

Figure 5.17 shows the results of the same approach in 3 random networks that have 5000, 10000 and 25000 peers, respectively. The approach shows the same performance in optimizing the relative intra-cluster similarity in these 3 networks. For the results of optimizing the clustering efficiency, however, significant differences can be observed. After 100 rewiring cycles, the clustering efficiency of about 0.95, 0.89, and 0.7 are achieved for the networks with 5000, 10000, and 25000 peers, respectively. For each peer in these three networks, on average, it has respectively 131, 262 and 656 peers whose distance is below or equal to the threshold $\theta = 0.5$. The resulting clustering efficiency thus implies that each peer can access about 124, 235, and 459 of these peers, respectively.

Besides, according to the previous simulation (Figure 5.6), we can observe that in the network with 25000 peers, if the clustering coefficients is calculated with $\gamma = 4$, the SA-based local search approach with Glauber dynamics can achieve a clustering efficiency 0.97. So if we use $\gamma = 4$ rather than $\gamma = 3$ to calculate the clustering efficiency in this subsection, we may achieve the similar clustering efficiency for these three networks. Furthermore, if we consider to use a larger number of short-range contacts, the resulting clustering efficiency could also be similar in these three networks.

Therefore, we can conclude that our approach allows each peer to access more similar peers as the network size and the number of similar peers increase. It exhibit a quite steady performance in both relative intra-cluster similarity and clustering efficiency.



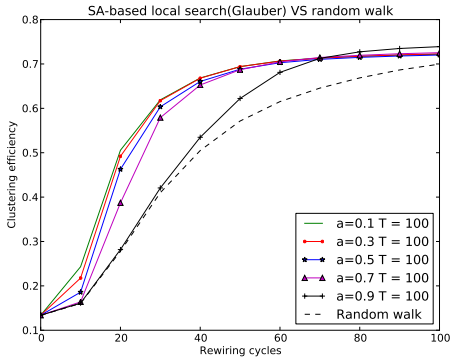
(a) Random network 1

(b) Random network 2

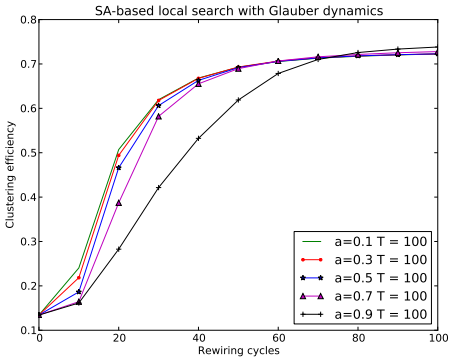
(c) Random network 3

(d) Random network 4

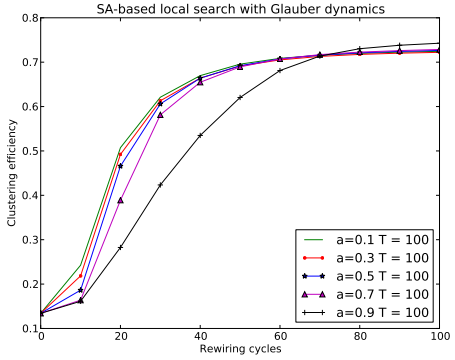
Figure 5.15: Relative intra-cluster similarity of SA-based local search with Glauber dynamics in different random network topology with 25000 peers (Random network topology 1 is the one we used to achieve the previous simulation results).



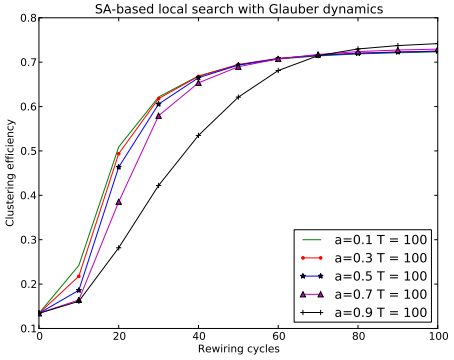
(a) Random network 1



(b) Random network 2

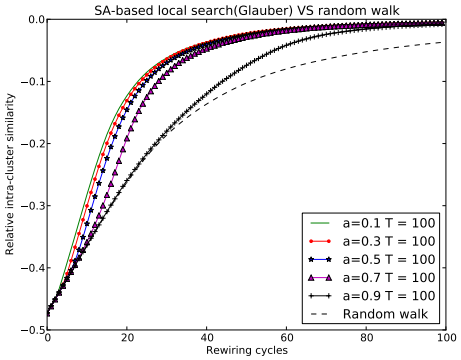


(c) Random network 3

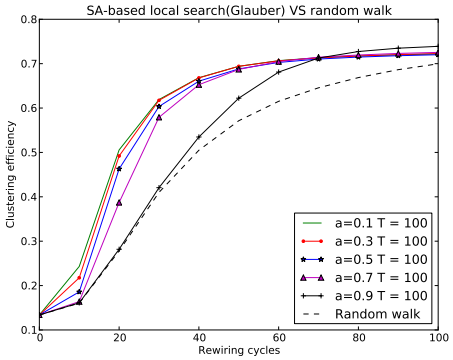


(d) Random network 4

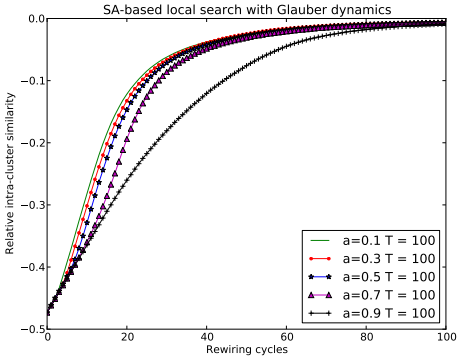
Figure 5.16: Clustering efficiency of SA-based local search with Glauber dynamics in different random network topology with 25000 peers (Random network 1 is the one we used to achieve the previous simulation results).



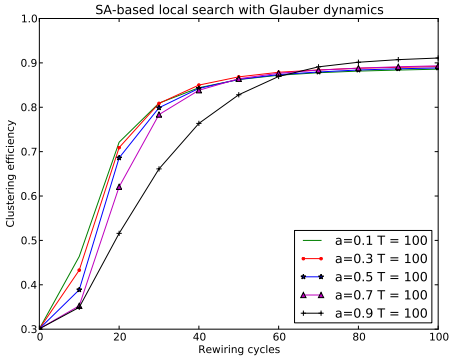
(a) Network with 25000 peers



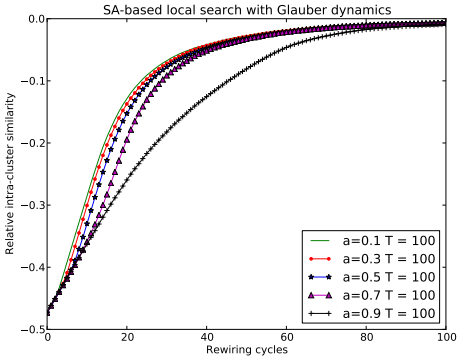
(b) Network with 25000 peers



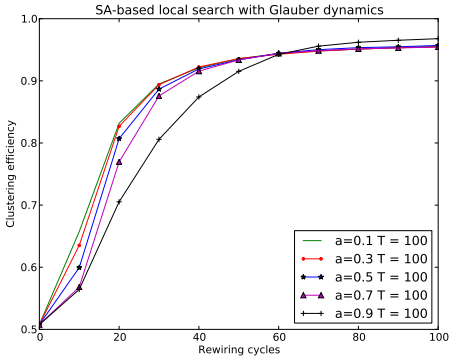
(c) Network with 10000 peers



(d) Network with 10000 peers



(e) Network with 5000 peers



(f) Network with 5000 peers

Figure 5.17: Relative intr-cluster similarity and clustering efficiency of SA-based local search with Glauber dynamics in random network topologies with different size (Network with 25000 peers is the one we used to achieve the previous simulation results).

5.5 Summary

In this chapter, the proposal of this thesis was validated by extensive simulation experiments. Random P2P networks were firstly generated, with each peer hosting a set of text documents in a single subject. Peer rewiring was then performed in the networks, aiming at transforming the random networks to SONs. The optimization progress of a SON was evaluated by tracing the evolution of relative intra-cluster similarity, clustering efficiency and the subsequent IR recall. Then new peers were simulated to join the network along the evolution of its topology. The peer rewiring behavior of the new peers was evaluated by the evolution of their intra-cluster similarity and clustering efficiency. The state-of-the-art approaches were implemented as baselines. In the end, experiments were made to study the effect of the parameters over the performance of peer rewiring approaches.

According to the simulation, we come to the following conclusions: (i) with an appropriate cooling schedule and Glauber dynamics, our SA-based local search has higher convergence speed comparing to the state of the art approaches. It uses less rewiring cycles than the approaches in the state of the art to generate the SONs with a certain quality. For example, the random walk takes 100 rewiring cycles to achieve a SON with relative intra-cluster similarity -0.03, our approach takes 40 rewiring cycles (Table 5.4); the random walk takes 100 rewiring cycles to achieve a SON with clustering efficiency 0.69, our approach takes 50 rewiring cycles to achieve that (Table 5.7). (ii) the simulation verifies our statement in Chapter 3: a high relative intra-cluster similarity does not necessarily imply a high clustering efficiency, although both of them are used to quantify the peer clusters in the network. The optimization of these two objects depends on the specific local search algorithm. For example, greedy walk achieves a higher relative intra-cluster similarity than random walk after 100 rewiring cycles, but it obtains lower clustering efficiency than random walk (Figure 5.1 and Figure 5.5); SA-based local search approach with Glauber dynamics, however, optimize both the intra-cluster similarity and clustering efficiency (Figure 5.2 and Figure 5.6), thanks to its appropriate usage of the random and greedy walk. (iii) SA-based local search with Metropolis dynamics does not outperform the state of the art approaches, because it always accepts good solutions without considering the improvements they make and hence does not have the advantage to efficiently discover better solutions. Instead, SA-based local search with Glauber dynamics has high probability to take better solutions, so its performance is better than the state of the art approaches. (iv) Since clustering efficiency can decrease the number of the hops a peer needs to access another similar peer by following the short-range links, a high clustering efficiency usually allows a

good IR performance within a peer cluster. However, its effect over the subsequent IR performance also depends on the specific query routing approach. Readers can refer to Figure 5.6 and Figure 5.9 to revisit the experimental proof. (v) SA-based local search with Glauber dynamics outperforms the other approaches in joining new peers. It spends less rewiring cycles to achieve a certain relative intra-cluster similarity and clustering efficiency, and obtain higher relative intra-cluster similarity and clustering efficiency after certain number of rewiring cycles (Figure 5.13 and Figure 5.14).

Our experimental study about the configurations and their effects on the performance of building SONs and IR demonstrates that: (i) it is necessary to define a distance threshold between each peer and its short-range contacts, because a proper threshold can control the quality of the similar peers a peer can access via the short-range links, and then affect the performance of target tasks like IR (Table 5.12); (ii) a proper combination of the number of short-range links s and the TTL of the query message k_q must be chosen in order to obtain acceptable IR performance (Table 5.13), while increasing the number of the long-range links after a certain value does not improve much of the performance in building SONs, if only the TTL of the walker and its walking strategy allow it to access all(most) of the peers in the network (Table 5.14); (iii) the performance of our SA-based approach remains the same for the networks with initially different random topology, according to the results in Figure 5.15 and Figure 5.16; (iv) our approach shows a high potential to allow peers to access more similar peers in the network as the network size and the number of similar peers increase, but more investigations are required in order to prove its scalability.

Chapter 6

Conclusions and Future Work

Contents

6.1 Conclusions 117

6.2 Future Work 119

6.2.1 Designing Adaptive Cooling Schedule 119

6.2.2 Refining Peer Profile and Similarity Measurement 120

6.2.3 Refining Neighborhood Exploration 121

6.2.4 Improving IR Performance within Peer Clusters 121

6.1 Conclusions

This thesis studied the task of building Semantic Overlay Networks (SONs) in unstructured P2P networks, for the application of P2P-IR. This task aims to cluster peers with semantically similar content (Chapter 1). It is a challenging task because no central coordinator or global structure exists to facilitate the clustering. Peers only have a limited local knowledge about their neighbors. With the local knowledge, peers are allowed to perform local operations to rewiring their connections to similar peers. These local operations make it difficult to globally cluster up all the similar peers in the network. Moreover, the dynamic behaviors of the peers require a robust rewiring mechanism which can efficiently maintain the network topology.

We identified and formalized peer rewiring as a repeated local search process performed by each peer. The peer periodically sends a walker to its neighborhood. The walker walks through the neighborhood via a certain local search strategy, collects information about the peers it explores, and returns with the information to its initiator. The collected information is used to update current connections to more similar peers (Chapter 3).

In order to better understand peer rewiring towards SONs, we recast peer rewiring as a decentralized local search solution to a combinatorial optimization problem: building SONs in which similar peers are clustered. Our optimization model reveals an observable gap between the combinatorial combination problem

and the decentralized local search solution: the local solution does not necessarily result in a global optimum topology. This motivates us to find a local solution with desirable properties that can lead to a global topology that is optimal or close to the optimum. Specifically, in order to better cluster the similar peers, while individual peers aims at rewiring their short-range links to achieve an optimized configuration of these links, they must play certain number of random walks properly to make connected the isolated similar peers (Chapter 3).

The traditional strategy to explore peers in a neighborhood is designed to be static. It does not consider the correlation between the strategy and the evolving network topology. In this thesis, we proposed an evolving walking strategy based on Simulated Annealing (SA), to consider the evolution of the network topology to improve the performance of peer rewiring (Chapter 4). Thanks to SA, a parameter called temperature was used to indicate the network topology state. A high temperature indicates an random network, while a low temperature indicates the emergence of peer clusters. With a high temperature, a random strategy was used to explore peers in the neighborhood to cluster up the isolated similar peers, while with a low temperature, more greedy strategy and less random strategy was used to speed up the clustering process and refine the peer connections. The strategy gradually changes as the temperature gradually decreases (as the network topology gradually evolves into a SON).

Our extensive simulations (Chapter 5) showed that the proposed approach can greatly improve the convergence time of building SONs as well as the IR performance in it. Specifically, we obtained the following conclusions:

(i) with an appropriate cooling schedule and Glauber dynamics, our SA-based local search shows higher convergence speed and results in higher quality of peer clusters. Specifically, our approach requires less rewiring cycles than the approaches in the state of the art to generate the SONs with a certain quality. For example, the random walk takes 100 rewiring cycles to achieve a SON with clustering efficiency 0.69, our approach takes 50 rewiring cycles to achieve it; our approach results in a clustering efficiency of 0.73 after 100 rewiring cycles, while random walk only achieves a clustering efficiency of 0.69.

(ii) properly accepting bad peers (or performing random walks) is important to perform the decentralized local search task in our study. It can decide if a decentralized local search algorithm can optimize both the relative intra-cluster similarity and the clustering efficiency in the object function. For example, greedy walk achieves a higher relative intra-cluster similarity than random walk after 100 rewiring cycles, but it obtains lower clustering efficiency than random walk; SA-based local search approach with Glauber dynamics, however, optimizes both the

intra-cluster similarity and clustering efficiency, thanks to its appropriate usage of the random and greedy walk.

(iii) SA-based local search with Metropolis dynamics does not outperform the state of the art approaches, because it always accepts good solutions without considering the improvements they make and hence does not have the advantage to efficiently discover better solutions. Instead, SA-based local search with Glauber dynamics has high probability to take better solutions, so its performance is better than the state of the art approaches.

(iv) Since clustering efficiency can decrease the number of the hops a peer needs to access another similar peer by following the short-range links, a high clustering efficiency usually allows a good IR performance within a peer cluster. However, its effect over the subsequent IR performance also depends on the specific query routing approach.

(v) SA-based local search with Glauber dynamics outperforms the other approaches in discovering similar peers for new peers. It spends less rewiring cycles to achieve a certain relative intra-cluster similarity and clustering efficiency for the new peers, and obtain higher relative intra-cluster similarity and clustering efficiency after a certain number of rewiring cycles.

Our experimental study about the configurations demonstrated that: (i) it is necessary to define a distance threshold between each peer and its short-range contacts, because a proper threshold can control the quality of the similar peers a peer can access via the short-range links, and then affect the performance of target tasks like IR; (ii) a proper combination of the number of short-range links s and the TTL of the query message k_q must be chosen in order to obtain acceptable IR performance, while increasing the number of the long-range links after a certain value does not improve much of the performance in building SONs; (iii) Our approach can build SONs from random networks, but how the networks are randomly configured does not affect the performance of our approach, if only the number of the connections for each peer is the same; (iv) our approach shows a high potential to allow peers to access more similar peers in the network as the network size and the number of similar peers increase, but more investigations are required in order to prove its scalability.

6.2 Future Work

6.2.1 Designing Adaptive Cooling Schedule

In our SA-based decentralized local search approach, the parameters for cooling schedule have been achieved by trial-and-error. This takes time and depends on

the specific network configuration. In other words, the same experimental effort is required in order to find the proper cooling schedule when SA-based decentralized local search is used in another network topology. An adaptive cooling schedule will be useful to avoid this effort.

Besides, an adaptive cooling schedule is necessary when more dynamic behaviors happen in the network, like peer leaving and changing their content. The frequency and the amount of these dynamic behaviors could heavily affect the network topology, and hence require a cooling schedule that is adaptive to these changes.

To achieve a proper adaptive cooling schedule, the cooling schedule must adjust the temperature's rate of decrease based on the information obtained during the algorithm's execution [Bertsimas 1993]. Some fast simulated annealing approaches in other applications can be studied [Ingber 1989, Ingber 1996] and adapted into our task.

6.2.2 Refining Peer Profile and Similarity Measurement

In this thesis, peer profile is designed to be represented as a set of topics. The similarity between two peer profiles is the Jaccard distance between two topic sets. However, more information can be integrated to generate a richer peer profile: (i) besides representing the peer profile as a set of topics, a real value can also be assigned to each topic of the profile, to quantify its contrition(weight) to represent the peer's content/interest. These values can be used to refine the similarity between two peer profiles. This idea is similar to using the term frequency instead of only terms to better compare the similarity of two documents [Turney 2010]; (ii) the topics can also be associated with some time-related information, such as when the topic emerges in the peer profile, when the user updates some documents about this topic. With the time-related information, we can determine if the topic is still an active topis in the peer profile, or if the topic emerges recently [Wang 2006]. This could be also useful to refine the similarity between two peer profiles.

With a richer peer profile explained above, a new similarity measurement can be designed. Obviously, it will not be a trivial task, because more information is involved. For example, if a topic in a peer profile A is quite new and with a low quantified contribution, it is not equivalent to the same topic in another profile B with long life span and a high quantified contribution. Moreover, the similarity measurement may not be symmetric: if B shares A's topics and these topics are active topics with high weight in B's profile, for profile A, profile B might be a very similar profile, because it has the potential to answer A's queries in terms of these topics; while for profile B, A may be not a very similar one, because the same topics are not important topics in A, and thus A does not show the advantage to

answer B's queries in terms of these topics. Although it is not a trivial task, a measurement based on the richer profiles can better evaluate the similarity between two peer profiles, and thus benefit the subsequent task of IR or collaborate filtering.

6.2.3 Refining Neighborhood Exploration

When peers rewire current connections to similar peers, they perform local search, which evolves from random walk to greedy walk, in their neighborhood in order to find the good peers. Random walk is used to search the neighborhood by randomly following the links, while greedy walk only follows the links that point to the most similar peer in the current local space. In other words, in this thesis, only profile similarity is considered when peers explore their neighborhood to rewiring current connections.

However, in a real P2P social network [Mani 2010, Durr 2012], rather than keeping the same number of connections, peers may have different number of connections according to the popularity of their interests. In this case, peers in the network are featured by another property: *peer centrality*. The peer centrality can be quantified as the number of the connections or the number of pathes that pass it [Freeman 1979]. A high peer centrality exhibits a strong capability to access other peers in the network [Freeman 1979].

If two of a peer's contacts have the same similarity to the peer but different centrality, the peer is probably able to find more similar peers through the contact with high centrality rather than through the contact with low centrality. Let's consider peer p_i sends a walker R_i to explore its neighborhood, and the walker has to choose between peer p_j and p_k as its next step. If p_j and p_k have the same similarity to p_i, the state of the art approaches as well as our proposal will allow the walker randomly to choose one of them as its next step, while a rational choice can be made based on the peers' centrality. If the walker chooses the peer with higher centrality as its next step, more similar peers could be found since the peer with higher centrality can access more peers in the network. Therefore, by integrating peer centrality into the operation of neighborhood exploration, building SONs is expected to be more efficient.

6.2.4 Improving IR Performance within Peer Clusters

We considered IR as the target application of SONs in this thesis. We focused on generating and maintaining high-quality SONs, and employed the most intuitive approach to perform IR within a cluster: the queries are flooded to the peers whose similarity to the query initiator is above a threshold. This can verify the quality

of the generated peer clusters. However, it is costly to flood a query to all the similar peers (in terms of a similarity threshold) within a cluster. In addition, the overall IR performance is low by using the intuitive IR, because the relevant documents may be also kept in the peers with a similarity below the threshold to the query initiator. Therefore, more refined query forwarding can be applied within a peer cluster. For example, peers can record the information of the queries they answered or forwarded, manage the information is a routing table, and then the table can be used to answer/forward the future queries more precisely [Kumar 2005b, Valdez 2010].

To take one more step forward, semantic query routing could be possible by inferring semantic relationship between terms. An initial proposal is outlined as follows: each peer records the queries it treats (forwarding and answering) as query feedback as well as their initiators. Peers can exchange their query feedback if necessary. With the query feedback, a term-peer frequency matrix can be achieved. The matrix provides the information of term co-occurrence in certain peers. If two terms appears almost in the same peers, a latent semantic relationship may exist between them. To infer latent semantic relationship between the terms based on query feedback, approaches in the field of social tagging could be introduced [Markines 2009]. The latent semantic relationship can be used for query routing or query extension.

Bibliography

[Aarts 2003] Emile HL Aarts and Jan Karel Lenstra. Local search in combinatorial optimization. Princeton University Press, 2003. (Cited on page 52.)

[Adamic 2003] Lada A Adamic, Rajan M Lukose and Bernardo A Huberman. *Local search in unstructured networks.* Handbook of graphs and networks, page 295, 2003. (Cited on page 18.)

[Balke 2005] Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski and Uwe Thaden. *Progressive Distributed Top-k Retrieval in Peer-to-Peer Networks.* In Proceedings of the 21st International Conference on Data Engineering, ICDE '05, pages 174–185, Washington, DC, USA, 2005. IEEE Computer Society. (Cited on pages 18, 72, 74 and 75.)

[Banaei-Kashani 2004] Farnoush Banaei-Kashani and Cyrus Shahabi. *SWAM: a family of access methods for similarity-search in peer-to-peer data networks.* In Proceedings of the thirteenth ACM international conference on Information and knowledge management, CIKM '04, pages 304–313, New York, NY, USA, 2004. ACM. (Cited on page 25.)

[Battiti 2009] Roberto Battiti, Franco Mascia and Mauro Brunato. *Reacting on the Annealing Schedule.* In Reactive Search and Intelligent Optimization, volume 45, pages 1–9. Springer US, 2009. (Cited on pages 53 and 69.)

[Bawa 2003] Mayank Bawa, Gurmeet Singh Manku and Prabhakar Raghavan. *SETS: search enhanced by topic segmentation.* In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pages 306–313, 2003. (Cited on pages 21 and 25.)

[Bender 2007] Matthias Bender, Tom Crecelius, Mouna Kacimi, Sebastian Michel, Josiane Xavier Parreira and Gerhard Weikum. *Peer-to-peer information search: Semantic, social, or spiritual?* IEEE Data Eng. Bull., vol. 30, no. 2, pages 51–60, 2007. (Cited on pages 73, 74, 79 and 99.)

[Bertier 2010] Marin Bertier, Davide Frey, Rachid Guerraoui, Anne-Marie Kermarrec and Vincent Leroy. *The gossple anonymous social network.* In Middleware 2010, pages 191–211. Springer Berlin Heidelberg, 2010. (Cited on pages 25, 26, 27, 30, 73 and 74.)

[Bertsimas 1993] Dimitris Bertsimas, John Tsitsiklis*et al.* *Simulated annealing*. Statistical Science, vol. 8, no. 1, pages 10–15, 1993. (Cited on pages 56, 58, 64 and 120.)

[BitTorrent 2013] BitTorrent. *http://www.bittorrent.com/*, 2013. (Cited on page 1.)

[Blei 2003] David M. Blei, Andrew Y. Ng and Michael I. Jordan. *Latent Dirichlet Allocation*. J. Mach. Learn. Res., vol. 3, pages 993–1022, 2003. (Cited on page 30.)

[Brooks 1995] SP Brooks and BJT Morgan. *Optimization using simulated annealing*. The Statistician, pages 241–257, 1995. (Cited on page 64.)

[Buford 2010] John F Buford and Heather Yu. *Peer-to-peer networking and applications: Synopsis and research directions*. In Handbook of Peer-to-Peer Networking, pages 3–45. Springer, 2010. (Cited on pages 12, 13 and 15.)

[Cholvi 2004] Vicent Cholvi, Pascal Felber and Ernst Biersack. *Efficient search in unstructured peer-to-peer networks*. European transactions on telecommunications, vol. 15, no. 6, pages 535–548, 2004. (Cited on pages 22, 25 and 26.)

[Cohen 2007] Edith Cohen, Amos Fiat and Haim Kaplan. *Associative search in peer to peer networks: Harnessing latent semantics*. Computer Networks, vol. 51, no. 8, pages 1861–1881, 2007. (Cited on page 23.)

[corpus 2011] Reuter corpus. *http://about.reuters.com/researchandstandards/corpus/*, 2011. (Cited on page 72.)

[Cramer 2004] C. Cramer, K. Kutzner and T. Fuhrmann. *Bootstrapping locality-aware P2P networks*. In Networks, 2004. (ICON 2004). Proceedings. 12th IEEE International Conference on, volume 1, pages 357–361 vol.1, 2004. (Cited on page 16.)

[Crespo 2002a] Arturo Crespo and Hector Garcia-Molina. *Routing indices for peer-to-peer systems*. In Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on, pages 23–32. IEEE, 2002. (Cited on page 73.)

[Crespo 2002b] Arturo Crespo and Hector Garcia-Molina. *Semantic Overlay Networks for P2P Systems*, 2002. (Cited on pages 1, 11, 17, 18, 19 and 49.)

[Darlagiannis 2005] Vasilios Darlagiannis. *Hybrid Peer-to-Peer Systems*. In Peer-to-Peer Systems and Applications, pages 353–366, 2005. (Cited on page 14.)

[Dhara 2010] Krishna Dhara, Yang Guo, Mario Kolberg and Xiaotao Wu. *Overview of structured peer-to-peer overlay algorithms.* In Handbook of Peer-to-Peer Networking, pages 223–256. Springer, 2010. (Cited on page 13.)

[Dorogovtsev 2001] Sergey N Dorogovtsev, José Fernando F Mendes and Alexander N Samukhin. *Giant strongly connected component of directed networks.* Physical Review E, vol. 64, no. 2, page 025101, 2001. (Cited on page 109.)

[Doulkeridis 2006] Christos Doulkeridis, Kjetil Nørvåg and Michalis Vazirgiannis. *The SOWES approach to P2P web search using semantic overlays.* In Proceedings of the 15th international conference on World Wide Web, pages 1027–1028. ACM, 2006. (Cited on page 21.)

[Doulkeridis 2007] Christos Doulkeridis, Kjetil Norvag and Michalis Vazirgiannis. *DESENT: Decentralized and distributed semantic overlay generation in P2P networks.* Selected Areas in Communications, IEEE Journal on, vol. 25, no. 1, pages 25–34, 2007. (Cited on pages 21 and 25.)

[Doulkeridis 2008] Christos Doulkeridis, Kjetil Nørvåg and Michalis Vazirgiannis. *Peer-to-peer similarity search over widely distributed document collections.* In Proceedings of the 2008 ACM workshop on Large-Scale distributed systems for information retrieval, pages 35–42, 2008. (Cited on pages 19, 21 and 25.)

[Doulkeridis 2010] Christos Doulkeridis, Akrivi Vlachou, C Doulkeridis, A Vlachou and M Vazirgiannis. *Distributed Semantic Overlay Networks*, 2010. (Cited on pages 1, 17, 73 and 74.)

[Draidi 2011] Fady Draidi, Esther Pacitti and Bettina Kemme. *P2Prec: a P2P recommendation system for large-scale data sharing.* In Transactions on large-scale data-and knowledge-centered systems III, pages 87–116. Springer, 2011. (Cited on pages 30 and 76.)

[Durr 2012] Michael Durr, Marco Maier and Kevin Wiesner. *An Analysis of Query Forwarding Strategies for Secure and Privacy-Preserving Social Networks.* In Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012), pages 535–542. IEEE Computer Society, 2012. (Cited on page 121.)

[Eberspächer 2005] Jörg Eberspächer and Rüdiger Schollmeier. *5. First and Second Generation of Peer-to-Peer Systems.* In Peer-to-Peer Systems and Applications, pages 35–56. Springer, 2005. (Cited on page 15.)

[Eisenhardt 2003] Martin Eisenhardt, Wolfgang Müller and Andreas Henrich. *Classifying Documents by Distributed P2P Clustering.* GI Jahrestagung (2), vol. 35, pages 286–291, 2003. (Cited on page 25.)

[Fletcher 2005] George HL Fletcher, Hardik A Sheth and Katy Börner. *Unstructured peer-to-peer networks: Topological properties and search performance.* In Agents and Peer-to-Peer Computing, pages 14–27. Springer, 2005. (Cited on page 13.)

[Freeman 1979] Linton C Freeman. *Centrality in social networks conceptual clarification.* Social networks, vol. 1, no. 3, pages 215–239, 1979. (Cited on page 121.)

[Garcia-Molina 2003] Hector Garcia-Molina and Arturo Crespo. *Semantic overlay networks for p2p systems.* Rapport technique, Stanford, 2003. (Cited on page 19.)

[Glover 1999] Fred Glover and Manuel Laguna. Tabu search. Springer, 1999. (Cited on page 53.)

[Henderson 2003] Darrall Henderson, Sheldon H Jacobson and Alan W Johnson. *The theory and practice of simulated annealing.* In Handbook of metaheuristics, pages 287–319. Springer, 2003. (Cited on page 55.)

[Ingber 1989] Lester Ingber. *Very fast simulated re-annealing.* Mathematical and computer modelling, vol. 12, no. 8, pages 967–973, 1989. (Cited on page 120.)

[Ingber 1996] Lester Ingber. *Adaptive simulated annealing (ASA): Lessons learned.* In Control and cybernetics. Citeseer, 1996. (Cited on page 120.)

[Ingber 2012] Lester Ingber, Antonio Petraglia, Mariane Rembold Petraglia, Maria Augusta Soares Machado*et al.* *Adaptive simulated annealing.* In Stochastic global optimization and its applications with fuzzy adaptive simulated annealing, pages 33–62. Springer, 2012. (Cited on page 64.)

[Jannotti 2000] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek and James W. O'Toole Jr. *Overcast: Reliable Multicasting with on Overlay Network.* In Proceedings of the 4th Conference on Symposium on Operating System Design & Implementation - Volume 4, OSDI'00, pages 14–14, 2000. (Cited on page 10.)

[Jin 2010] Xing Jin and S-H Gary Chan. *Unstructured Peer-to-Peer Network Architectures.* In Handbook of Peer-to-Peer Networking, pages 117–142. Springer, 2010. (Cited on page 15.)

[Kalogeraki 2002] Vana Kalogeraki, Dimitrios Gunopulos and D. Zeinalipour-Yazti. *A local search mechanism for peer-to-peer networks.* In Proceedings of the eleventh international conference on Information and knowledge management, CIKM '02, pages 300–307, 2002. (Cited on page 13.)

[Ke 2010] Weimao Ke and Javed Mostafa. *Scalability of findability: effective and efficient IR operations in large information networks.* In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, pages 74–81. ACM, 2010. (Cited on pages 73 and 74.)

[Kim 2008] Jae Kyeong Kim, Hyea Kyeong Kim and Yoon Ho Cho. *A user-oriented contents recommendation system in peer-to-peer architecture.* Expert Systems with Applications, vol. 34, no. 1, pages 300–312, 2008. (Cited on page 27.)

[Kim 2011] Sungsu Kim. *Semantic overlay network for peer-to-peer hybrid information search and retrieval.* In Proceedings of 2011 IFIP/IEEE International Symposium on Integrated Network Management(IM), pages 430–437, 2011. (Cited on page 73.)

[Kirkpatrick 1983a] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. *Optimization by simulated annealing.* Science, vol. 220, pages 671–680, 1983. (Cited on pages 53 and 57.)

[Kirkpatrick 1983b] Scott Kirkpatrick, D. Gelatt Jr. and Mario P Vecchi. *Optimization by simmulated annealing.* science, vol. 220, no. 4598, pages 671–680, 1983. (Cited on page 53.)

[Klampanos 2004] Iraklis A Klampanos and Joemon M Jose. *An architecture for information retrieval over semi-collaborating peer-to-peer networks.* In Proceedings of the 2004 ACM symposium on Applied computing, pages 1078–1083. ACM, 2004. (Cited on pages 21 and 25.)

[Kolen 1994] Antoon Kolen and Erwin Pesch. *Genetic local search in combinatorial optimization.* Discrete Applied Mathematics, vol. 48, no. 3, pages 273–284, 1994. (Cited on page 42.)

[Koloniari 2008] Georgia Koloniari and Evaggelia Pitoura. *Recall-based cluster reformulation by selfish peers.* In Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on, pages 200–205. IEEE, 2008. (Cited on pages 22 and 27.)

[Kumar 2005a] Abhishek Kumar, Jun Xu and Ellen W Zegura. *Efficient and scalable query routing for unstructured peer-to-peer networks.* In INFOCOM

2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, volume 2, pages 1162–1173. IEEE, 2005. (Cited on page 73.)

[Kumar 2005b] Abhishek Kumar, Jun Xu and Ellen W Zegura. *Efficient and scalable query routing for unstructured peer-to-peer networks.* In 24th IEEE International Conference on Computer Communications (INFOCOM 2005), volume 2, pages 1162–1173, 2005. (Cited on pages 79, 99 and 122.)

[Kurve 2013] Aditya Kurve, Christopher Griffin, David J Miller and George Kesidis. *Optimizing cluster formation in super-peer networks via local incentive design.* Peer-to-Peer Networking and Applications, pages 1–21, 2013. (Cited on page 19.)

[Levandowsky 1971] Michael Levandowsky and David Winter. *Distance between sets.* Nature, vol. 234(5), pages 34–35, 1971. (Cited on page 30.)

[Levin 2009] David Asher Levin, Yuval Peres and Elizabeth Lee Wilmer. Markov chains and mixing times. American Mathematical Soc., 2009. (Cited on page 57.)

[Lewis 2004] David D. Lewis, Yiming Yang, Tony G. Rose and Fan Li. *RCV1: A New Benchmark Collection for Text Categorization Research.* J. Mach. Learn. Res., vol. 5, pages 361–397, December 2004. (Cited on page 73.)

[Li 2008] Mei Li, Wang-Chien Lee, Anand Sivasubramaniam and Jing Zhao. *SSW: A Small-World-Based Overlay for Peer-to-Peer Search.* IEEE Transactions on Parallel and Distributed Systems, vol. 19, pages 735–749, 2008. (Cited on pages 24, 25 and 26.)

[Linari 2006] Alessandro Linari and Gerhard Weikum. *Efficient peer-to-peer semantic overlay networks based on statistical language models.* In Proceedings of the international workshop on Information retrieval in peer-to-peer networks, P2PIR '06, pages 9–16, New York, NY, USA, 2006. ACM. (Cited on pages 24 and 31.)

[Löser 2004] A Löser, F Naumann and W Siberski. *Semantic overlay clusters within super-peer networks.* Information Systems, 2004. (Cited on page 19.)

[Löser 2007] Alexander Löser, Steffen Staab and Christoph Tempich. *Semantic social overlay networks.* IEEE Journal on Selected Areas in Communications, vol. 25, no. 1, pages 5–14, 2007. (Cited on pages 24, 25 and 26.)

[Lourenço 2010] Helena R Lourenço, Olivier C Martin and Thomas Stützle. *Iterated local search: Framework and applications.* In Handbook of Metaheuristics, pages 363–397. Springer, 2010. (Cited on page 53.)

[Lua 2005] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, Steven Lim*et al.* *A survey and comparison of peer-to-peer overlay network schemes.* IEEE Communications Surveys and Tutorials, vol. 7, no. 1-4, pages 72–93, 2005. (Cited on pages 10 and 13.)

[Lv 2002] Qin Lv, Pei Cao, Edith Cohen, Kai Li and Scott Shenker. *Search and replication in unstructured peer-to-peer networks.* In Proceedings of the 16th international conference on Supercomputing, pages 84–95. ACM, 2002. (Cited on page 18.)

[Mani 2010] M. Mani, Anh-Minh Nguyen and N. Crespi. *SCOPE: A prototype for spontaneous P2P social networking.* In 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pages 220–225, 2010. (Cited on page 121.)

[Marin 2009] M. Marin, V. Gil-Costa and C. Hernandez. *Dynamic P2P Indexing and Search Based on Compact Clustering.* In Similarity Search and Applications, 2009. SISAP '09. Second International Workshop on, pages 124–131, Aug 2009. (Cited on pages 18 and 73.)

[Markines 2009] Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominik Benz, Andreas Hotho and Gerd Stumme. *Evaluating Similarity Measures for Emergent Semantics of Social Tagging.* In Proceedings of the 18th International Conference on World Wide Web, WWW'09, pages 641–650, 2009. (Cited on page 122.)

[Merz 2006] Peter Merz and Steffen Wolf. *Evolutionary local search for designing peer-to-peer overlay topologies based on minimum routing cost spanning trees.* Parallel Problem Solving from Nature-PPSN IX, 2006. (Cited on page 48.)

[Metropolis 1953] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller. *Equation of State Calculations by Fast Computing Machines.* Journal of Chemical Physics, vol. 21, pages 1087–1092, June 1953. (Cited on page 56.)

[Nourani 1998] Yaghout Nourani and Bjarne Andresen. *A comparison of simulated annealing cooling strategies.* Journal of Physics A: Mathematical and General, vol. 31, no. 41, 1998. (Cited on page 58.)

[Ohnishi 2012] Kei Ohnishi, M Köppen, K Yoshida and Y Oie. *Evolutionary P2P Networking for Realizing Adaptive Networks.* Advances in Intelligent Modelling, 2012. (Cited on page 48.)

[Papapetrou 2007] Odysseas Papapetrou, Wolf Siberski, W-T Balke and Wolfgang Nejdl. *Dhts over peer clusters for distributed information retrieval.* In Advanced Information Networking and Applications, 2007. AINA'07. 21st International Conference on, pages 84–93. IEEE, 2007. (Cited on pages 17 and 74.)

[Papapetrou 2010] Odysseas Papapetrou, Wolf Siberski and Wolfgang Nejdl. *PCIR: Combining DHTs and peer clusters for efficient full-text P2P indexing.* Computer Networks, vol. 54, no. 12, pages 2019–2040, 2010. (Cited on pages 17, 73, 74 and 75.)

[Parreira 2007] Josiane Xavier Parreira, Sebastian Michel and Gerhard Weikum. *p2pDating: Real life inspired semantic overlay networks for Web search.* Inf. Process. Manage., vol. 43, no. 3, pages 643–664, 2007. (Cited on pages 7, 23, 24, 49, 52 and 78.)

[PeerSoN 2012] PeerSoN. *http://www.peerson.net/*, 2012. (Cited on page 1.)

[Penzo 2008] Wilma Penzo, Stefano Lodi, Federica Mandreoli, Riccardo Martoglia and Simona Sassatelli. *Semantic peer, here are the neighbors you want!* In Proceedings of the 11th international conference on Extending database technology: Advances in database technology, EDBT '08, pages 26–37, 2008. (Cited on pages 24 and 27.)

[Podnar 2007] Ivana Podnar, Martin Rajman, Toan Luu, Fabius Klemm and Karl Aberer. *Scalable peer-to-peer web retrieval with highly discriminative keys.* In Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, pages 1096–1105. IEEE, 2007. (Cited on pages 17, 18, 73, 74 and 75.)

[Raftopoulou 2008a] Paraskevi Raftopoulou and Euripides G M Petrakis. *iCluster: a self-organizing overlay network for P2P information retrieval.* In Proceedings of the IR research, 30th European conference on Advances in information retrieval, ECIR'08, pages 65–76, 2008. (Cited on pages 5, 7, 12, 23, 30, 31, 33, 37, 39, 49, 54 and 73.)

[Raftopoulou 2008b] Paraskevi Raftopoulou and Euripides GM Petrakis. *A measure for cluster cohesion in semantic overlay networks.* In Proceedings of the 2008 ACM workshop on Large-Scale distributed systems for information retrieval, pages 59–66. ACM, 2008. (Cited on page 40.)

[Raftopoulou 2008c] Paraskevi Raftopoulou, Euripides GM Petrakis, Christos Tryfonopoulos and Gerhard Weikum. *Information retrieval and filtering over self-organising digital libraries.* In Research and Advanced Technology for Digital Libraries, pages 320–333. Springer, 2008. (Cited on pages 23 and 26.)

[Raftopoulou 2009a] Paraskevi Raftopoulou. *Rewiring Peer Connections over Semantic Overlay Networks.* PhD thesis, Technical University of Crete, 2009. (Cited on pages 2, 27, 51, 52 and 54.)

[Raftopoulou 2009b] Paraskevi Raftopoulou, Euripides GM Petrakis and Christos Tryfonopoulos. *Rewiring strategies for semantic overlay networks.* Distributed and Parallel Databases, vol. 26, no. 2-3, pages 181–205, 2009. (Cited on page 11.)

[Raftopoulou 2010] Paraskevi Raftopoulou and Euripides G M Petrakis. *Peer rewiring in semantic overlay networks under churn.* In Proceedings of the 2010 international conference on the move to meaningful internet systems, OTM'10, pages 573–581, 2010. (Cited on pages 11, 27, 74 and 78.)

[Ratnasamy 2001] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp and Scott Shenker. *A Scalable Content-addressable Network.* SIGCOMM Comput. Commun. Rev., vol. 31, no. 4, pages 161–172, 2001. (Cited on pages 13 and 16.)

[Reynolds 2003] Patrick Reynolds and Amin Vahdat. *Efficient Peer-to-peer Keyword Searching.* In Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware, Middleware '03, pages 21–40, 2003. (Cited on page 17.)

[Risson 2006] John Risson and Tim Moors. *Survey of research towards robust peer-to-peer networks: search methods.* Computer networks, vol. 50, no. 17, pages 3485–3521, 2006. (Cited on page 17.)

[Rutenbar 1989] R A Rutenbar. *Simulated annealing algorithms: an overview.* Circuits and Devices Magazine, IEEE, vol. 5, no. 1, pages 19–26, 1989. (Cited on page 55.)

[Sakaryan 2003a] German Sakaryan and H Unger. *Topology Evolution in P2P Distributed Networks.* Applied Informatics, 2003. (Cited on page 48.)

[Sakaryan 2003b] German Sakaryan and Herwig Unger. *Influence of the Decentralized Algorithms on Topology Evolution.* Design, Analysis, and Simulation of Distributed Systems (DASD, pages 12–18, 2003. (Cited on page 49.)

[Sakaryan 2004] German Sakaryan. *A content-oriented approach to topology evolution and search in peer-to-peer systems.* PhD thesis, University of Rostock, 2004. (Cited on page 49.)

[Schmitz 2004] Christoph Schmitz. *Self-Organization of a Small World by Topic.* In LWA, pages 303–310. Citeseer, 2004. (Cited on pages 5, 7, 24, 30, 31, 33, 49, 51, 52 and 54.)

[Schollmeier 2001] Rüdiger Schollmeier. *A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications.* In Peer-to-Peer Computing, 2001. Proceedings. First International Conference on, pages 101–102. IEEE, 2001. (Cited on page 16.)

[Sedmidubsky 2008] Jan Sedmidubsky, Stanislav Barton, Vlastislav Dohnal and Pavel Zezula. *Adaptive approximate similarity searching through metric social networks.* In Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on, pages 1424–1426, 2008. (Cited on pages 22 and 25.)

[Skobeltsyn 2006] Gleb Skobeltsyn and Karl Aberer. *Distributed cache table: efficient query-driven processing of multi-term queries in P2P networks.* In Proceedings of the international workshop on Information retrieval in peer-to-peer networks, pages 33–40. ACM, 2006. (Cited on pages 73 and 74.)

[Sripanidkulchai 2003] K. Sripanidkulchai, B. Maggs and Hui Zhang. *Efficient content location using interest-based locality in peer-to-peer systems.* In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, volume 3, pages 2166–2176, 2003. (Cited on pages 18, 19, 22, 25, 26 and 31.)

[Stoica 2001] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications.* In Proceedings of the ACM SIGCOMM '01 Conference, pages 149–160, San Diego, California, August 2001. (Cited on pages 13 and 16.)

[Tang 2003a] Chunqiang Tang, Zhichen Xu and Sandhya Dwarkadas. *Peer-to-peer information retrieval using self-organizing semantic overlay networks.* In Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pages 175–186. ACM, 2003. (Cited on pages 11, 18 and 19.)

[Tang 2003b] Chunqiang Tang, Zhichen Xu and Mallik Mahalingam. *pSearch: Information retrieval in structured overlays.* ACM SIGCOMM Computer Com-

munication Review, vol. 33, no. 1, pages 89–94, 2003. (Cited on pages 17, 18, 19 and 74.)

[Tempich 2004] Christoph Tempich, Steffen Staab and Adrian Wranik. *Remindin': semantic query routing in peer-to-peer networks based on social metaphors.* In Proceedings of the 13th international conference on World Wide Web, pages 640–649. ACM, 2004. (Cited on pages 22, 25, 26 and 39.)

[Tigelaar 2012] Almer S Tigelaar, Djoerd Hiemstra and Dolf Trieschnigg. *Peer-to-peer information retrieval: An overview.* ACM Transactions on Information Systems (TOIS), vol. 30, no. 2, page 9, 2012. (Cited on pages 11 and 19.)

[Traversat 2003] Bernard Traversat, Ahkil Arora, Mohamed Abdelaziz, Mike Duigou, Carl Haywood, Jean christophe Hugly, Eric Pouyoul and Bill Yeager. *Project JXTA 2.0 Super-Peer Virtual Network.* Sun Microsystem White Paper, 2003. (Cited on page 15.)

[Triantafillou 2003] Peter Triantafillou, Chryssani Xiruhaki, Manolis Koubarakis and Nikos Ntarmos. *Towards High Performance Peer-to-Peer Content and Resource Sharing Systems.* In CIDR, 2003. (Cited on page 21.)

[Turney 2010] Peter D. Turney and Patrick Pantel. *From Frequency to Meaning: Vector Space Models of Semantics.* J. Artif. Int. Res., vol. 37, no. 1, pages 141–188, January 2010. (Cited on page 120.)

[Valdez 2010] GC Valdez. *A self-adaptive ant colony system for semantic query routing problem in p2p networks.* Computación y Sistemas, vol. 13, no. 4, pages 433–448, 2010. (Cited on page 122.)

[Voulgaris 2007] Spyros Voulgaris, Maarten van Steen and Konrad Iwanicki. *Proactive gossip-based management of semantic overlay networks.* Concurrency and Computation: Practice and Experience, vol. 19, no. 17, pages 2299–2311, 2007. (Cited on pages 5, 7, 11, 23, 26, 27, 31, 49, 54, 74 and 78.)

[Wang 2003] Chonggang Wang and Bo Li. *Peer-to-peer overlay networks: A survey.* Department of Computer Science, The Hong Kong University of Science and Technology, Hong Kong, 2003. (Cited on page 9.)

[Wang 2006] Xuerui Wang and Andrew McCallum. *Topics over Time: A non-Markov Continuous-time Model of Topical Trends.* In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pages 424–433, New York, NY, USA, 2006. ACM. (Cited on page 120.)

[Wiśniewski 2011] Michał Wiśniewski and Krzysztof Walkowiak. *Evolutionary algorithm for P2P multicasting network design problem.* In Hybrid Artificial Intelligent Systems, pages 182–189. Springer, 2011. (Cited on page 48.)

[Wolf 2009] Steffen Wolf and Peter Merz. *Iterated local search for minimum power symmetric connectivity in wireless networks.* In Evolutionary Computation in Combinatorial Optimization, pages 192–203. Springer, 2009. (Cited on page 48.)

[YaCy 2011] YaCy. *http://yacy.de/en/index.html*, 2011. (Cited on page 1.)

[Zhang 2007] Rongmei Zhang and Y Charlie Hu. *Assisted peer-to-peer search with partial indexing.* Parallel and Distributed Systems, IEEE Transactions on, vol. 18, no. 8, pages 1146–1158, 2007. (Cited on page 74.)

Acronyms

P2P Peer-to-Peer

IR Information Retrieval

P2P-IR Peer-to-Peer Information Retrieval

C/S Client/Server

SON Semantic Overlay Network

SA Simulated Annealing

TTL Time to live

Notations

$G = \langle P, L \rangle$	the network with a set of peers P and a set of links L
P	peers in the network
L	links in the network
N	the size of the network
P	the set of peers in the network
s	number of short range links for each peer
l	number of long range links for each peer
p_a	a specific peer a
p_b	a specific peer b
p_c	a specific peer c
p_i	any peer i
p_j	any peer j
p_k	any peer k
D_i	document collection in p_i
$l_{i,j}$	any link from p_i to p_j
P_{short}	a set of peers, referring to short-range contacts
P_{long}	a set of peers, referring to long-range contacts
P^i_{short}	a set of peers, referring to the short-range contacts of p_i
P^j_{short}	a set of peers, referring to the short-range contacts of p_j
P^r_{short}	a set of peers, referring to the short-range contacts of p_r
p_{max}	the short-range contact with maximum distance
d_{max}	the distance to p_{max}
P^i_{long}	a set of peers, referring to the long-range contacts of p_i

$topic_i$ the ID of any topic

θ the threshold for intra-cluster distance

$IntraDis$ the name of the function for intra-cluster distance

Dis the name of the function for distance between peers

R_i rewiring message (walker) send by p_i

p_r host peer of the rewiring message

p_e any explored peer

$P^i_{collected}$ a set of peers, referring to the peers collected for p_i

P'_r possible host peers of the rewiring message in the next step

p'_r possible host peer of the rewiring message in the next step

n_r number of peers a rewiring message is sent to in parallel

q_i query initiated by p_i

Q_i query message initiated by p_i

γ hop limit for the metric of clustering effficiency

S_0 initial solution

S current solution

S' neighboring solution

S^0_i initial solution for peer i

S_i current solution for peer i

S'_i neighboring solution for peer i

T_0 initial temperature for cooling schedule

t current annealing step

T_t current temperature

T_{t+1} temperature for next step

E energy of current solution

E' energy of new solution

$S_{optimal}$ optimal solution

a parameter for exponential cooling schedule

$tMax$ maximum steps of simulated annealing

$eMax$ maximum value of the object function

$rTime$ time to relax at each temperature

Publications

1. Sylvie Cazalens, Yulian Yang, Sylvie Calabretto and Esther Pacitti, Sur l'utilisation de LDA en RI pair-à-pair, Actes du XXIXème Congrès INFORMED, pp. 1-8, Paris, France, 29-31 May 2013

2. Yulian Yang, Sylvie Calabretto and Lionel Brunn, Centrality based peer rewiring in semantic overlay networks (S), IEEE International Conference on Research Challenges in Information Science, Paris, pp. 1-6, Paris, France, 2013

3. Yulian Yang, Sylvie Calabretto and Lionel Brunn, Information Retrieval in Self-organizing Networks, International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA) PhD seminar, Lugano, Italy, 2012

4. Yulian YANG, Sylvie Calabretto and Lionel Brunn, Semi-structured semantic overlay for information retrieval in self-organizing networks, World Wide Web (WWW) PhD Symposium, pp. 203-208, Lyon, France, 2012

5. Yulian Yang, Semantic Information Retrieval over P2P Network, Rencontres Jeunes Chercheurs en Recherche d'Information (RJCRI), COnférence en Recherche d'Information et Applications(CORIA), pp. 391-396, Avignon, France, March 2011