TR-2007-18:Optimizing Acquaintance Selection in a PDMS

Jian Xu
Computer Science Department
University of British Columbia
Vancouver, Canada
xujian@cs.ubc.ca

Rachel Pottinger
Computer Science Department
University of British Columbia
Vancouver, Canada
rap@cs.ubc.ca

ABSTRACT

In a Peer Data Management System (PDMS), autonomous peers share semantically rich data. For queries to be translated across peers, a peer must provide a mapping to other peers in the PDMS; peers connected by such mappings are called *acquaintances*. To maximize query answering ability, a peer needs to optimize its choice of acquaintances. This paper introduces a novel framework for performing acquaintance selection. Our framework includes two selection schemes that effectively and efficiently estimate mapping quality. The "one-shot" scheme clusters peers and estimates the improvement in query answering based on cluster properties. The "two-hop" scheme, estimates using locally available information at multiple rounds. Our empirical study shows that both schemes effectively help acquaintance selection and scale to PDMSs with large number of peers.

1. INTRODUCTION

A Peer Data Management System (PDMS) (e.g., [12, 2, 21]) combines the flexibility of ad-hoc sharing of information in a peer-to-peer network with the richer semantics of a database. In a PDMS, each source is assumed to have a database to share, rather than just exchanging files. This allows the users of the PDMS to exchange semantic-rich data, rather than only exchanging simple files. Since these peers are autonomous, they are assumed to have their own schemas. To solve this problem, PDMSs require semantic mappings between the various schemas.

Consider the example PDMS in Figure 1. In response to a recent earthquake, four cell phone companies (CHEAP CELL PHONES, CELL PHONES FOREVER, CELL PHONE LAND, CELL PHONE EASY), a land-based telephone company (LAND LINES R US), an electric company (ELECTRIC COMPANY) and a cable company (HAPPY CABLE) have quickly formed a PDMS to share data to see the global problems for their customers and shared infrastructures. To establish basic connectivity, they have created a small number of mappings (shown as lines between peers in Figure 1). Similarities may vary substantially between peers. For example, CHEAP CELL PHONES has much in common with CELL PHONES FOREVER, and a lot (but less) in common with LAND LINES R US. HAPPY CABLE and ELECTRIC COMPANY share under-water pipes for their wires, and ELECTRIC COMPANY and LAND LINES R US share utility poles. For the network shown in Figure 1, a query from HAPPY CABLE will need to be translated through the two CELL PHONE EASY mappings before it can be processed at ELECTRIC COMPANY. Peers that are directly connected to each other through such a semantic mapping are called *acquaintances*. e.g., HAPPY CABLE's only acquaintance is CELL PHONE EASY.

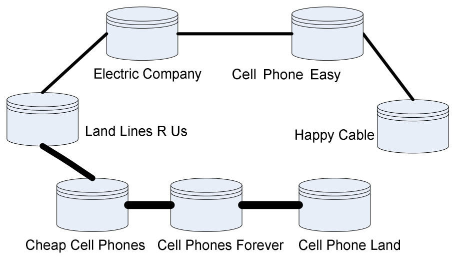


Figure 1: An example of a PDMS and semantically mapped peers

Much research has focused on answering queries in a PDMS by composing the semantic mappings (e.g., [11, 15, 26]), which has been shown to be computationally complex and often undecidable. Additionally, there is considerable work on decreasing the cost to construct such mappings (see [8] for a recent survey). Since the mappings are inherently difficult to determine — being able to determine them fully automatically would require solving all of Artificial Intelligence — the best that can be done is to create such mappings semi-automatically. Due to these factors, the scale of a PDMS is limited; unlike a typical file-exchange peer-to-peer network, which can have thousands of peers, PDMSs are usually formed by no more than several hundred peers.

While semi-automatic schema matching techniques decrease the costs, it remains too expensive to create mappings between a newly joined peer and all other peers in the system. In our example, though creating a mapping between HAPPY CABLE and all other peers will yield the optimal ability to answer queries, HAPPY CABLE may not have the resources to do so — particularly in a disaster management situation, where time is critical. Thus, it is very important for a peer to choose acquaintances to maximize its ability/usage of the PDMS. In our continuing example, Cell Phone Easy is in a poor position because it lacks a cell phone company as its acquaintance. Although queries from Cell Phone Easy can still be translated along the established mappings, query answering is limited, e.g., cell-phone specific queries may be blocked because the peers on route to the other cell phone peers lack cell-phone specific schema elements.

Note that the best new acquaintance for a peer may not be the candidate that has the best potential mapping quality with it. Consider our example in Figure 1; assume that the mapping between Cell Phone Land and Cheap Cell Phones is predicted to have a higher mapping quality than the mapping between Cell Phone Land and Cell Phone Easy. If Cell Phone Land is considering creating a new mapping, its best choice may be Cell Phone Easy instead of Cheap Cell Phones because it already has a high quality semantic path to Cheap Cell Phones through the mappings via Cell Phones Forever while a query will need to take 5 hops of translation to Cell Phone Easy. The key for a selection criteria is the **extra benefit** when a candidate is chosen as acquaintance.

Because the queries that can be translated across peers vary greatly depending on which peers are selected as acquaintances, it is imperative that a peer adding a new acquaintance can tell which peers are likely to be of the greatest help on query answering, if chosen as acquaintances, without fully creating the mappings involved. We call this the *acquaintance selection problem*: given an existing PDMS, and a peer p, which may already have some acquaintances in the PDMS, how can p choose new acquaintance(s) to maximize its ability to translate queries?

As shown above, there are two aspects to the acquaintance selection problem: (1) the ability of the new acquaintance to help answer queries must be estimated and (2) it must be estimated how well queries can be answered without the proposed acquaintance. This paper describes two schemes to the acquaintance selection problem. The first acquaintance selection scheme, the "one-shot" scheme classifies peers into a set of clusters and selects a new acquaintance based on discovered clustering property, though the best choice may not be in the same cluster. e.g. In example 1, suppose all cell phone companies are clustered together, Cell Phone Land may choose Electric Company, which is not in its cluster, over Cheap Cell Phones because it already has a good query answering path with the latter and the benefit of creating an extra mapping with Cheap Cell Phones is therefore low. The "one-shot" scheme pre-processes all peers in the PDMS in one pass and thus the selection process afterwards virtually takes no extra time to estimate direct mapping potential. The second solution, namely the "two-hop" scheme, explores the network in multiple rounds and performs acquaintance selection using the information available locally at each round. Whereas one-shot is quite efficient when we know roughly the number of clusters of peers, two-hop can be used when this information is unavailable. In addition, the two-hop theme is more adaptive, it refines estimations more easily when new information becomes available. Our empirical study shows that both schemes effectively help acquaintance selection and scale to large number of peers.

We make the following specific contributions:

- We propose a general acquaintance selection framework, including the operations required.
- We propose a clustering based "one-shot" scheme to quickly estimate the quality of a potential mapping.
- We propose a "two-hop" scheme that is more flexible than the one-shot scheme. Two-hop helps a peer explore the network in multiple rounds and make acquaintances selection decision at each round.
- We empirically evaluate the effectiveness and efficiency of the two acquaintance selection schemes.

The paper is organized as follows. Sections 2 and 3 describe background and related work. Section 4 describes the acquaintance selection framework and identifies the primitive operations needed. Section 5 describes the two selection schemes and analyzes them in detail. We present our empirical evaluation in Section 6. Finally, we conclude the paper and describe future work in Section 7.

2. PRELIMINARIES

2.1 PDMS query answering and mappings

While schema mappings can have many representations and mapping composition methods can vary, our technique depends on neither a specific mapping type, nor on a specific composition algorithm. However, readers wanting a specific format for examples should consider the system in [15]: a schema mapping from data source A to data source B, denoted as M_{A_B} is a set of mapping formulas of the form $Q_A(\overline{X}) \subseteq Q_B(\overline{X})$, where Q_A and Q_B are conjunctive queries over R_A and R_B respectively. For example, Cell Phone Land has two relations

```
    User (ID, Name, Age, DOB)
    PhoneNum (ID, Number,ContractRef)
```

and Cell Phone Forever has

```
    Customer(Name,Number,DOB,Address,Email)
```

A mapping formula between the two schemas is

```
Customer(Name,Number,DOB) =
    User(ID,Name,DOB),PhoneNum(ID,Number)
```

We also adopt the definition of mapping composition in [15]: a composition of M_{A_B} and M_{B_C}, denoted as $M{*}_{A\_C} = Comp(M_{A\_B}, M_{B\_C})$ is a mapping to directly translate query written in schema A to schema C without the need of intermediate schema B. Note that a composed mapping is not equivalent to a direct mapping (e.g. mapping M_{A_C}) that the quality of the composed mapping is affected both by the base mappings (e.g. M_{A_B} and M_{B_C}) and the mapping composition algorithm.

We assume that the PDMS is unstructured (i.e., the network is not required to conform to a particular topology). A peer queries the PDMS by flooding its query to the network. Queries are re-written along established schema mappings in the network and are processed on every peer. For example in Figure 1, a query q from Cheap Cell Phones will be re-written to q' in Cell Phones Forever's schema using the mapping between them. Then Cell Phone Forever will both answer q' and forward q' to Cell Phone Land by re-writing it into q'' using its other mapping. Please refer to [15] for a detailed description of query rewriting.

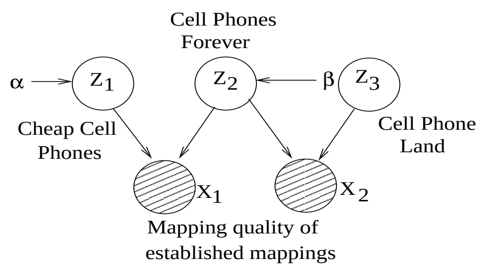


Figure 2: An example of graphical model

2.2 Graphical model

The graphical model is widely used in AI research to represent the dependencies of random variables and hyper parameters in a system. We use it to represent the one-shot estimator in Section 5.5. A graphical model uses nodes to represent random variables and an arrow between them to denote a dependency relationship. If a variable in the system can be observed, which means its value is determined, we shadow it (e.g., X_1 and X_2 in Figure 2). An observed variable is also called "evidence". Latent variables, which have values that are never observed, are shown as unfilled nodes (e.g., Z_1 and Z_2 in Figure 2). Typically, the observed variables depend on the latent variables. The goal of the graphical model is to use the observed variables to find the values of the latent variables. For example, Figure 2 shows a graphical model representation for part of our system in Figure 1. The top three variables Z_1, Z_2 and Z_3 represent the schemas of Cheap Cell Phones, Cell Phones Forever and Cell Phone Land respectively. Variable X_1 is the quality of the mapping between Cheap Cell Phones and Cell Phones Forever. This quality depends on the latent variables Z_1 and Z_2 which indicates their schema characteristics.

In most cases, unobserved variables will follow a certain distribution which are controlled by a set of parameters. These parameters are called "hyper parameters" and are represented by pointing to the variable using Greece characters. In example 2, Z_1 follows Poisson distribution with parameter α and Z_2's distribution parameter is β. So given a graphic model representation of a system, we can know the dependencies among the variables. In example 2, we can know that X_1 depends on Z_1 and Z_2 but is independent to Z_3.

For more a detailed description of graphical models, see [3].

2.3 MAP estimates

Maximum a posteriori (MAP) estimation is used widely in statistical analysis to estimate an un-observed variable based on some observed data [3]. Consider the example shown in Figure 2; the MAP estimate for the values of latent variables Z_1 and Z_2 can be written as

$$(\widehat{Z_1, Z_2})_{MAP} = \arg\max_{Z_1, Z_2} P(X_1, X_2|Z_1, Z_2)P(Z_1, Z_2)$$

where $P(Z_1, Z_2)$ is the prior knowledge on the (joint) distribution of Z_1 and Z_2. The benefit in using MAP as opposed to other estimates such as maximal likelihood estimator (MLE) or a full Baysian treatment is that it enables prior knowledge to be applied to the estimation. The use of the mode of the distribution instead of the mean (expectation) frees the estimator to compute the full full distribution of the variable being inferred. As we will see in our one-shot estimator, MAP estimates opens the opportunity for scalable local search.

2.4 Expectation Maximization (EM)

Expectation maximization (EM), is a family of algorithms used to infer parameters in a system [3]. Take the system shown in Figure 2 for example. We want to infer the values of α and β in the system, given the value of X_1 and X_2 as observed evidence. The EM algorithm consists of two steps. The "E-step" computes $P(Z_1, Z_2|X_1, \alpha_0, \beta_0)$, $P(Z_2|X_1, X_2, \beta_0)$ using a initial parameter α_0 and β_0. Then the "M-step" updates values of α and β according to some update function. (In most cases to maximize the expected (log)likelihood value). The EM algorithm repeats the above E-step and M-step until the parameter(s) converge. As we will see in Section 5.5, our one-shot estimator uses a modification of an EM algorithm to obtain the value of the clustering parameters and the MAP estimate of the latent variable.

3. RELATED WORK

People have long noted that selecting good neighbors can reduce networking costs in structured (e.g., Chord [25], Pastry [22]) and unstructured (e.g., Napster, Gnutella) P2P networks. Both [19, 28] discuss peer selection and grouping strategies to lower networking cost. Selecting peers to form groups and exploiting the locality opportunity (by getting as much as possible from nearby neighbors) is also studied in works [17, 27, 24, 30, 5]. In [14], clustering information is used to reduce large scale flooding in the network. Some structured P2P works [6, 7, 4] study neighbor selection based on proximity information to enable efficient routing.

An approach to improve query routing quality for information retrieval on a clustering-based architecture is reported in [13]. A recent work [16] discusses efficient query routing for PDMSs in the WISDOM project; queries are passed from one to another following certain semantic routing indices, so that a good balance between query answering quality and networking cost can be achieved.

All of the above, while also focusing on the peer selection problem, do not address the problem we have now. The goal of the previous neighbor selection techniques is improving networking costs/efficiency, while our focus is maximizing the semantic querying ability of a peer.

There are also a number of related works in machine learning and pattern recognition areas. Some works on pairwise clustering algorithms [23, 18, 10] have proposed methods to learn the pattern of a data set given pairwise distance information, which is directly related to our one-shot acquaintance selection scheme. Another pairwise clustering approach reported in [20] also uses EM in clustering. Our approach differs substantially from theirs on the basic assumptions of variable distributions, the function to optimize and the detail optimizing method, which theirs was developed for motion-segmentation applications. The works [23, 1] suggest looking into longer paths in the network than only the pairwise relations. This motivates our development of the two-hop acquaintance selection scheme.

4. THE ACQUAINTANCE SELECTION FRAMEWORK

We now introduce a new general framework for efficient acquaintance selection as defined in Section 1. For example, consider helping Cell Phone Easy in Figure 1 choose a new acquaintance to bolster its query answering capabilities. At its least precise, the framework consists of two layers of abstraction as depicted in Figure 3. The estimator layer consists of the estimators that are used by the estimate operation in the upper layer to compute the selection criteria adopted by a certain selection scheme. For the selection criteria identified in Section 1, we need two estimators; one estimating direct mapping quality and the other estimating the current mapping impact. The operation layer consists of mechanisms for both collecting information that is needed by the estimators and combining their estimations into the final acquaintance selection.

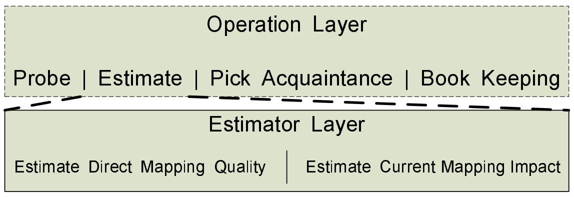

Operation Layer
Probe | Estimate | Pick Acquaintance | Book Keeping
Estimator Layer
Estimate Direct Mapping Quality
Estimate Current Mapping Impact

Figure 3: The acquaintance selection framework

The operations in the operation layer are the general actions that a peer will perform during the acquaintance selection process. However, depending on the specific selection scheme a peer uses, the actions performed and the estimators chosen may be different.

During acquaintance selection, the host peer will perform the 4 operations in this layer, and the **estimate** operation will use the estimators in estimator layer to compute the selection criteria. We describe in detail these two layers in the next two subsections.

4.1 The operation layer

1. **Probe** : The host peer collects information about other peers in the PDMS and mappings among them.
2. **Estimate** : The host peer estimates the benefit of mappings to candidate peers. In this step, it uses the estimators and computes the selection criteria.
3. **Pick Acquaintance**: The host peer picks one or more peers as acquaintances and establishes mappings with them. The quality measure of these newly established mappings is then computed on the host peer. Note that the estimate step only *estimates* the quality of these mappings. It is only after the mappings are fully built that a peer can compute the true mapping quality.
4. **Book Keeping**: The host peer keeps track of the information disseminated to other peers. The book keeping operation ensures no duplicated or redundant information is transmitted.

4.2 The estimator layer

The estimators generally depend on the acquaintance selection scheme used by the peers. For the schemes proposed in this paper, the estimators are categorized into two classes. The *Estimate Direct Mapping Quality* class contains those that estimate the direct mapping quality. Those in the *Evaluate Current Mapping Impact* class estimate the impact of existing query answering paths to the host peer.

Next we describe two acquaintance selection schemes that fit in this framework. We use "selection scheme" or "scheme" as short forms to "acquaintance selection scheme" wherever appropriate.

5. ACQUAINTANCE SELECTION SCHEMES

This section describes the one-shot and the two-hop selection schemes in detail. Both schemes use the mapping quality metric in Sections 5.1 and the selection criteria in Section 5.2. In Section 5.3, we describe how to compute the current aggregate mapping. Section 5.4 shows how to derive a fast approximation for estimating its quality, which is used by both selection schemes as the current mapping impact estimator. The one-shot scheme is then described in Section 5.5, and the two-hop scheme follows in Section 5.6.

5.1 Mapping and mapping quality metric

To decide which peer to pick as an acquaintance, the host peer needs to estimate the quality of the mapping, if established, to a candidate peer. Because the goal of a mapping is to allow query translation, one primary factor is the number of attributes that are mapped from the source schema to the target schema. While a sophisticated quality metric could be chosen, this paper proposes a quality metric that serves as a first approximation: it measures the fraction of attributes that are mapped. Formally:

Definition 1 (Mapping Quality Metric). *Let $sch(i)$ denote peer i's schema and $|sch(i)|$ be the number of attributes in $sch(i)$. Let x_{i_j} be the distinct number of attributes in $sch(i)$ that appear in mapping M_{i_j}. Then the mapping quality $S(M_{i\_j})$ is defined as $S(M_{i\_j}) = \frac{x_{i\_j}}{|sch(i)|}$* □

The mapping quality is normalized in $[0, 1]$, which, as we will see in Section 5.6 provides convenience. This quality metric can be used on both direct mappings between two peers and composed mappings. Note that it is by definition asymmetric. i.e., $S(M_{i\_j}) \neq S(M_{j\_i})$; quality $S(M_{i\_j})$ is measured at peer i's perspective. The set of all mapping quality in form of $S(M_{i\_\cdot})$ roughly measure peer i's query translating/answering ability with the existing mappings.

We now define the "current aggregate mapping" which is used by the current mapping impact estimator. The current aggregate mapping, as its name suggests, can be regarded as a virtual mapping that takes into consideration all the existing query translation paths from the host peer to a target peer in the PDMS. To define it properly, we first define a **cordless path**[1].

Definition 2 (Cordless Path). *In a graph $G(V, E)$, a cordless path p from vertex i to j is a path that satisfies*

1. ***(simple)**: p is acyclic*
2. ***(shortest)**: i, j with any vertex subset of p do not form a path from i to j* □

Figure 4 shows an example of cordless paths in a graph. The three solid grey paths are cordless paths from s to t while the two dashed paths are not (e.g., the upper path (s, a, b, c, t) is disqualified because of the cord (s,b)).

Observe the triangle (s, a, b) in the example and suppose we have mapping composition $M^*_{s\_b} = Comp(M_{s\_a}, M_{a\_b})$.

[1]Thanks to David Kirkpatrick for suggesting this name

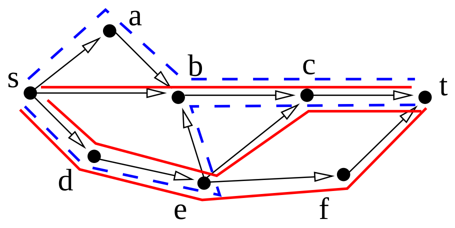


Figure 4: An example of cordless paths: (s, b, c, t)**,** (s, d, e, c, t)**,** (s, d, e, f, t) **are cordless paths from** s **to** t**.**

We say M_{s_b} **dominates** $M^*_{s\_b}$ if all the mapping rules in $M^*_{s\_b}$ appear in M_{s_b}. As mapping composition is an information lossy operation, we assume that a direct mapping between two peers will always dominate a composed mapping. Then, it is easy to see that if we compose mappings on all the cordless paths and union these composed mappings, we will get a minimal, dominating mapping rule set. It is minimal in the sense that if we miss any cordless path we could possibly lose mapping rules that can be used for query answering. Therefore, when computing current mapping impact, we can simply only consider the cordless paths.

Hence, we define the "Current Aggregate Mapping" from peer i to peer j as the mapping created by the union of composed mappings on all cordless paths from i to j. Formally:

Definition 3 (Current Aggregate Mapping). *The current aggregate mapping from peer i to peer j is defined as*

$$CM_{i\_j} = \bigcup_{p \in P(i,j)} (Comp(p))$$

where $P(i, j)$ is the set of cordless paths from i to j and $Comp(p)$ is the mapping composed from all mappings along path p. □

Note that mapping composition (Comp) itself can be an involved operation [11]. So even though we have reduced the path set to only cordless paths, computing CM explicitly is non-trivial. To resolve this, we instead estimate $Comp$ when we apply the mapping quality metric to CM.

With the current aggregate mapping defined, we can measure the mapping impact of a candidate peer to a host peer: the impact of peer j to host peer i can be seen as the mapping quality of the current aggregate mapping from i to j.

Now we are ready to describe the acquaintance selection criteria. In Section 5.2, we rely on the definition of the current aggregate mapping quality, but we defer describing how to compute it to Section 5.3.

5.2 The acquaintance selection criteria

As the example in Section 1 shows, using only the direct mapping quality as the selection criteria is insufficient. A peer's goal is to maximize its query answering ability, thus it should choose acquaintances that maximize the benefit to its query answering ability. This benefit can be quantified as the difference between the direct mapping quality $S(M_{i\_j})$ and the current aggregate mapping quality $S(CM_{i\_j})$.

Hence, we want a peer in the PDMS to choose an acquaintance that maximizes its query answering benefit. Using the concepts we have just defined, the selection criteria is

Definition 4 (selection criteria). *Peer i selects j as its acquaintance if and only if $S(M_{i\_j}) - S(CM_{i\_j}) \geq S(M_{i\_j'}) - S(CM_{i\_j'})\ \forall j' \neq j$, where $S(\cdot)$ is the mapping quality defined in Definition 1 and CM in Definition 3.* □

and we call $S(M_{i\_j}) - S(CM_{i\_j})$ the "criteria value" for j.

To use this selection criteria on a candidate peer j, the host peer i needs to compute both $S(M_{i\_j})$ and $S(CM_{i\_j})$. We first describe how to efficiently estimate $S(CM_{i\_j})$ between a host peer and all its candidate peers.

5.3 Computing CM for all candidates

By the definition of CM (Definition 3), computing $S(CM_{i\cdot})$ for all peer i's candidates requires that i first discovers the cordless path (Definition 2) set from itself to the candidate peers. We formalize this path finding problem as follows.

Definition 5 (Cordless path set discover problem). *Given a (directed) graph $G(V, E)$ where V and E represent vertex and edge sets respectively, and a source vertex $s \in V$, the cordless path set discover problem computes the cordless path set $P(s, j)$ for all $j \in V, j \neq s$.* □

We now present an algorithm we call the "CP finding algorithm" that solves the cordless path set discover problem. The algorithm creates works by calculating the cordless path set by adding each vertex v into a vertex set S and performing processing on the verticies. The algorithm terminates when all the vertices are added into S. We will show how the cordless path sets can be updated during this process.

We arbitrarily choose one vertex v from V and add it to S. If $(s, v) \in E$, then we store a path (s, v) on v. All length 1 paths are cordless paths. Suppose at some stage, a number of vertices have been added and the cordless paths from s to all $v \in S$, involving only vertices in S, are computed and stored on v. Now we show that we can add in a new vertex $k \in V - S$ to S and correctly update $P(s, v)$. We first compute the cordless paths from s to k that only involve vertices from $S \cup \{k\}$. We call these paths **"**$P(s, k)$ **w.r.t.** $S \cup \{k\}$**"**.

Let $V^S_{k-} = \{v | (v, k) \in E, v \in S\}$, the vertices in S that have an edge with k. It is apparent that $P(s, k)$ w.r.t $S \cup \{k\}$ is a subset of $\bigcup_{v \in V^S_{k-}} (P(s, v) \diamond (v, k))$, where $\diamond$ is an operator that appends edge (v, k) to every path in $P(s, v)$ to form a path set from s to k. Now observe that if two vertices $u, v \in V^S_{k-}$ where u appears in some path $p \in P(s, v)$, then $\{p\} \diamond (v, k)$ is not a cordless path because edge (u, k) is a shortcut. Similarly, we can argue that for $v \in V^S_{k-}$ and $p \in P(s, v)$ if $\nexists u \in V^S_{k-}$ and $u \in p$, then the new path $\{p\} \diamond (v, k)$ is a cordless path, i.e., it is in $P(s, k)$ w.r.t. $S \cup \{k\}$.

To quickly determine if a path can be extended to form a new cordless path, we store with each computed p in $P(s, v)$ a bit vector f_p of length $|V|$. Let $f_p[i] = 0$ indicate that vertex i **cannot** be extended to i on this path. We initialize each $f_p[i]$ to 1 to indicate that it may be able to be extended. Each time a path $p \in P(s, v)$ is to be extended to some vertex k, v will first check $f_p[k]$. If $f_p[k] = 1$ then path $p' = p \diamond (v, k)$ is added to $P(s, k)$ and $f_{p'}$ is set as following. First p' cannot extend to any vertex that p cannot extend to, so $f_{p'}[i] = 0$ if $f_p[i] = 0$. Second, p' cannot be extended to v or any of v's neighbor so $f_{p'}[i] = 0 \forall i \in V_{v+} = \{t | (v, t) \in E, t \in V\}$ and $f_{p'}[v] = 0$.

Now we must update the existing cordless path sets $P(s, v)$ w.r.t. S so that after updating, we get cordless path sets $P(s, v)$ w.r.t $S \cup \{k\}$ for all $v \in S$. Observe that all the existing paths in $P(s,v)$ are still valid cordless paths because none of them involves k; therefore, k does not create any shortcuts. Therefore we only need to consider paths that are extended from $P(s,k)$ w.r.t $S \cup \{k\}$, which we have just obtained. This can be done easily by performing the same test on f_p that we performed before for $p \in P(s,k)$ for $v \in V_{k+}$, where $V_{k+}^S = \{v|(k,v) \in E, v \in S\}$. Also, when a new path is added, the corresponding f_p vector is set. Figure 5 shows the procedure of adding k to S.

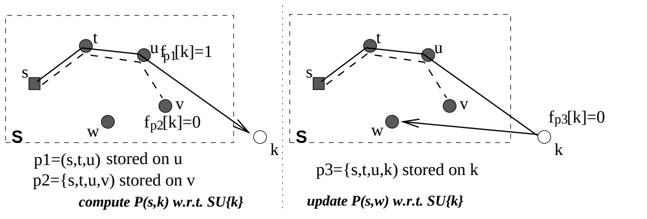


Figure 5: The process of adding in a new vertex k to the partial set S (shown by the dashed box) and computing $P(s,i)$ w.r.t $S \cup \{k\}$ for all i in $S \cup \{k\}$.

THEOREM 5.1. *The CP finding algorithm solves the cordless path set discover problem in $O(h)$ time for a (directed) graph G with $|E| = O(|V|)$, where h is the total number of cordless paths, assuming that selecting a path with $f_p[k] = 1$ for a cordless path set $P(s,v)$ takes constant time.*

PROOF. Computing $P(s,k)$ for a newly added vertex k requires only finding extendable paths in $P(s,v)$ for $v \in V_{k-}^S$. The number of such $P(s,v)$ is a constant amortized across all vertices, given that $|E| = O(|V|)$. Adding a new path to $P(s,k)$ requires two operations. First it requires extending a valid path, which takes constant time. Setting f_p for this new path involves finding one vertex's neighbors, which is also a constant under the $|E| = O(|V|)$ assumption. Therefore, all paths can be computed in $O(h)$ time. □

Note that h is the total number of cordless **paths**. For example in Figure 5, $p1$ and $p2$ are counted as two paths. A hash table on each $P(s,v)$ can be used to quickly find the paths that can be extended to a certain vertex k.

Here are some corollaries from the above theorem that are useful in computing $S(CM)$.

COROLLARY 5.1. *CM for all candidates can be computed in $O(h)$ time, where h is the total number of cordless paths.* □

COROLLARY 5.2. *When Graph G is extended by adding a new vertex v and a constant number of new edges connecting v with other vertices, then the P set can be updated in $O(h_{new})$ time where h_{new} is the number of newly formed cordless paths.* □

Corollary 5.3 immediately follows.

COROLLARY 5.3. *When a new peer and its mappings are discovered, CM for all candidates can be updated in $O(h_{new})$ time where h_{new} is the number of newly discovered cordless paths.* □

5.4 Max-min approximation for $S(CM)$

To estimate the current aggregate mapping and its quality (Definition 3), we use a "max-min" approximation to simplify the mapping composition ($Comp$) procedure. It assumes that number of attributes mapped in the aggregate mapping CM_{i_j} is the maximal number of mapped attributes in all mappings each composed along a cordless path from peer i to j, and the number of attributes mapped in such a composed mapping, is the minimal number of mapped attributes in all the mappings on this path. Let $x_{i\_j}^p$ denote the number of attributes in schema i mapped in the composed mapping along path p, to j. The current aggregate mapping quality, $S(CM_{i\_j})$, is max-min approximated following Definition 1, as

$$S(CM_{i\_j}) = \max_{p \in P(i,j)} x_{i\_j}^p / |sch(i)|$$

where $x_{i\_j}^p$ is estimated along path p using

$$x_{i\_j}^p = \min_{(a,b) \in p} x_{a\_b}$$

where x_{a_b} is the number of mapped attributes as defined in Definition 1.

We agree that this is a very rough approximation for a real mapping composition procedure that can be quite different and involved. However, this simplified procedure is sufficient for us to evaluate the performance of the proposed selection schemes without depending on a particular composition scheme. We will call this estimator the "max-min estimator". The complexity of the max-min estimator can easily be verified to follow Corollaries 5.1 and 5.3 by observing that max, min can be computed with the discovery of the cordless paths.

Next we describe in detail two selection schemes which differs mainly in their direct mapping quality estimators.

5.5 The one-shot selection scheme

We now present a selection scheme that we call the "one-shot scheme". As with all selection schemes, the one-shot scheme chooses its high-level structure from the operation layer (Section 4.1). In the one-shot scheme, the host peer first probes the PDMS to collect quality information for already established mappings. This information is referred to as the "topology" due to its analogy to a graph with peers as vertices and reported mapping quality as edges. The one-shot selection scheme uses a direct mapping quality estimator called the **one-shot estimator** in the estimate step. The one-shot estimator works by classifying peers in the PDMS into a number of clusters using the quality information from established mappings. Direct mapping quality estimations are then made using the properties of the discovered clusters. It identifies valuable candidates in *one pass* over the available information and thus is named "one-shot".

Assume that N peers in the PDMS need to be classified into C clusters. The estimator's goal is to find the best clustering so that pairwise mapping quality between two peers in the same cluster is high, while that between peers from different clusters is low. The one-shot estimator needs to know the number of clusters to classify the peers into. In many cases this information is available from the domain knowledge of the PDMS. (our two-hop selection scheme (Section 5.6) is developed for scenarios where this information is unavailable.) The challenge of the one-shot estimator is to discover best the cluster assignments using the limited established mapping quality information from the PDMS.

5.5.1 *The one-shot estimator in detail*

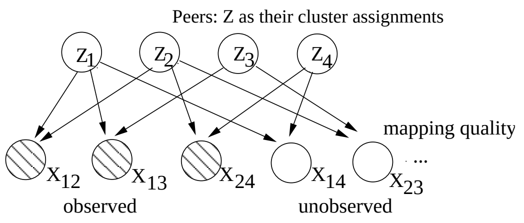


Figure 6: Variables and dependencies in one-shot estimator

The variables and their dependencies in the one-shot estimator are shown in Figure 6, where nodes are divided into two rows. The upper row consists of the peers with independent random variables Z_i representing their cluster assignments. The second row shows the pairwise mapping between every two peers; X_{ij} represents their mapping quality. Mapping quality is determined by a pair of peers. Some mappings are already established, so their quality is observed by the host peer (e.g., X_{12}, X_{13}, X_{24}). Other mappings are not established, and thus their quality must be estimated (e.g., X_{14} and X_{23}). The model assumes that the mapping quality between two peers, given their cluster assignments $Z_i = a$ and $Z_j = b$, follows Gaussian distribution with parameters $(\mu_{ab}, \delta^2_{ab})$. Depending on the number of clusters, C, a set of $|C| \times |C|$ Gaussian distributions is used to characterize the system. We use θ to denote these parameters with $\theta_{ij} = (\mu_{ij}, \delta^2_{ij})$.

The estimator's task is to infer both the cluster assignments $Z = (Z_1, \dots, Z_N)$ and the Gaussian parameters θ for all peers. We choose to search for the MAP estimates (Section 2.3) of Z and θ, formally:

$$\widehat{(Z, \theta)}_{MAP} = \arg\max_{Z,\theta} P(X|Z, \theta)P(Z)$$

The fact that θ values are continuous but Z is discrete makes it hard to optimize them together. To resolve this, we conduct a 2-phase optimization using a modified EM (Section 2.4): we first pick a θ and find the MAP estimates of Z under this fixed θ; then we optimize θ with Z value fixed. The two steps iterates until Z and θ converge.

After Z and θ have converged, the one-shot estimator estimates direct mapping quality between the host peer i and a candidate peer j. It returns the mean $\mu_{Z_i Z_j}$ as the mapping quality estimation. Next we describe how the MAP estimates of Z and θ can be computed.

5.5.2 *Local search on Z for MAP estimates*

In the first phase, we search for $\hat{Z}_{MAP}$ with a fixed θ. The function to optimize, as described above, is

$$f(Z) = P(X|Z, \theta)P(Z)$$

and the MAP estimates for Z, with θ fixed, is $\hat{Z}_{MAP} = \arg\max_Z f(Z)$ where X is the set of observed mapping quality. By the independence of Z_i, $f(Z)$ can be factorized and we study $\log f(Z)$ (they lead to the same MAP estimates) written as

$$\log f(Z) = \sum_{i,j} (\log N(x_{i,j}|\theta_{Z_i,Z_j}) + \log P(Z_i)P(Z_j)).$$

for all (i, j) with $x_{i,j} \in X$. Note that θ is fixed in this phase; the only variable is Z.

Let $Z^{[n]}$,$P^{[n]}(Z)$ and $\theta^{[n]}$ denote the Z, $P(Z)$ and θ in iteration n of the EM algorithm. The above formula can be transformed to an update function w.r.t the iterations.

$$\begin{aligned} Z^{[n+1]} &= \arg\max_Z \log f(Z) \\ &= \arg\max_Z \sum_{i,j} (\log N(x_{i,j}|\theta^{[n]}_{Z_i,Z_j}) + \log P^{[n]}(Z_i)P^{[n]}(Z_j)) \end{aligned}$$

with $P^{[n+1]}(t) = \frac{N_t}{N}$ where N_t is the number of $Z_i^{[n]}$'s that are valued t and is initialized as $P^{[1]}(t) = \frac{1}{C}$ for all t.

Finding $\hat{Z}_{MAP}$ (i.e., the cluster assignment) requires exhaustively searching all possible assignments; this is infeasible for a search space as big as C^N. We conduct a segmented local search that speeds up this procedure. Figure 7 shows an example of segmented local search.

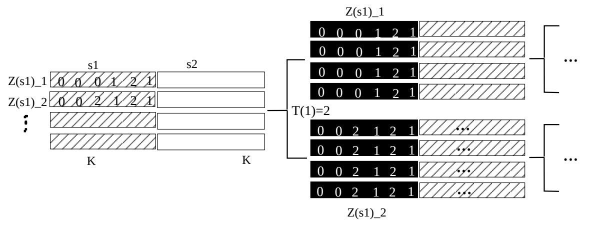


Figure 7: An example of segmented local search: Shadowed segments are active, white ones are segments yet to be searched, and black segments are fixed assignments carried over from the search on the previous segment. K is the segment length. It first searches all C^K assignments of $Z(s_1)$. Only the $T(1) = 2$ "best" assignments are carried over to the next step where each branch searches s_2 for $Z(s_1 \cup s_2)$ with carried over $Z(s_1)$.

The segmented local search algorithm works as follows: first, it picks a constant K as the segment length and breaks Z into $t = \lceil N/K \rceil$ segments $s_1..s_t$. It then performs segment by segment local search. Let $Z(s_i)$ denote the assignment of s_i. Starting with s_1, it searches each segment using exhaustive search, i.e., it tries all C^K assignments for the segment. For s_i, it computes $f(Z(\cup_{j \le i}(s_j)))$ with assignments on $\cup_{j<i}(s_j)$ fixed on one of the carried over assignments from the search on s_{i-1}. For each s_i, which has search space $S(i) = C^K * T(i-1)$, it keeps the best $T(i) = \log S(i)$ assignments to carry over to the search on segment s_{i+1} We define $T(0) = 1$.

The segmented local search finishes when all segments are processed and the assignment that yields the maximal $f(Z)$ is used in the second phase to optimizing θ, as described in the following section.

5.5.3 *Computing MAP estimates for θ*

We use the Z assignment found in the last step to compute $\hat{\theta}_{MAP}$. Still, the goal is to maximize $P(X|Z, \theta)P(Z)$. The updating function for θ can be written as

$$\theta^{[n+1]} = \arg\max_{\theta} \sum_{i,j} \log(P(x_{ij}|\theta_{Z_i^{[n]},Z_j^{[n]}})) + \log P^{[n]}(Z_i^{[n]})P^{[n]}(Z_j^{[n]})$$

Note that in this phase, Z is fixed and the only variable is θ. Solving (taking the derivatives and setting it equal to 0) under the assumption of Gaussian distribution,

$$P(x_{ij}|\theta_{Z_i^{[n]},Z_j^{[n]}}) \sim N(\mu_{Z_i^{[n]},Z_j^{[n]}}, \delta_{Z_i^{[n]},Z_j^{[n]}})$$

results in

$$\mu_{p,q}^{[n+1]} = \frac{1}{N_{p,q}^{[n]}} \sum_{Z_i^{[n]}=p, Z_j^{[n]}=q} x_{i,j}$$

$$\delta_{p,q}^{2[n+1]} = \frac{1}{N_{p,q}^{[n]}} \sum_{Z_i^{[n]}=p, Z_j^{[n]}=q} (x_{i,j} - \mu_{p,q}^{[n+1]})$$

where $N_{p,q}^{[n]}$ is the number of $x_{i,j}$'s with $Z_i^{[n]} = p$, $Z_j^{[n]} = q$.

In other words, μ and δ^2 are updated using the sample mean and standard deviation. If one cluster receives too few peers, the corresponding $(\mu_{p,q}, \delta_{p,q}^2)$ is set to a prior value, e.g., $(0.7, 0.03)$ if $p = q$ and $(0.3, 0.03)$ otherwise.

5.5.4 Convergence of iterations

The one-shot estimator iteratively optimizes Z and θ until they converge. In both steps of optimizing Z and θ, the value of $f(Z)$ is guaranteed to monotonically increase. Because $f(Z)$ is bounded, the iterations always converge by requiring as a convergence condition that the increment of $f(Z)$ is smaller than a threshold for a number of consecutive iterations.

5.5.5 Analysis of the one-shot estimator

The time complexity of the one-shot estimator in each iteration comes from the two optimization procedures. While it is easy to see that optimizing θ can be done in $O(N)$ time, searching for $\hat{Z}_{MAP}$ also finishes in $O(N)$ time.

The time complexity on searching for $\hat{Z}_{MAP}$ can be broke into two parts. One is the cost of searching through the assignments and the other is the cost of computing $f(Z)$ for each (partial) assignment. With the segmented local search described in section 5.5.2, we have the following lemma.

LEMMA 5.1. *Given N peers to classify into C clusters, if the segment length is set to K, The total number of assignments that the segmented local search tests is $O(C^K N \log C)$.*

PROOF. According to the segmented local search algorithm, the number of assignments of each segment of length $K$is $P = C^K$. Let $S(i)$ denote the search space when the searching is on segment i and $T(i)$ be the number of assignments get carried to the next segment. Then we have $S(1) = P, T(1) = logP$ and $S(t) = P \cdot T(t-1), T(t-1) = \log S(t-1)$ for $t > 1$. Hence,

$$S(t) = P \log S(t-1) = P \log P + P \log \log(S(t-2))$$

expanding $S(t)$ one more step we have

$$S(t) = P \log P + P \log[\log P + \log(\log S(t-3))]$$

because segments have uniform length, the term $\log(\log S(t-3))$ can be approximated by $\log\log(\mu C^K) < (1+\epsilon)\log K$, for small constants μ,ϵ. Note that $K > C$ we have,

$$\begin{aligned} S(t) &= P \log P + P \log[K \log C + (1+\epsilon)\log K] \\ &< C^K K \log C + C^K \log[K \log K + (1+\epsilon)\log K] \\ &= C^K K \log C + C^K [\log(K + 1 + \epsilon) + \log\log K] \\ &< C^K (K \log C + (1+\epsilon')\log K) \end{aligned}$$

where ϵ' is a small constant. Sum up for all $\lceil N/K \rceil$ segments, we get an upper bound of the total complexity

$$\sum_{i=1}^{\lceil N/K \rceil} S(i) = C^K [\log C + (1+\epsilon')\frac{\log K}{K}] N$$

with $\log C > (1+\epsilon')\frac{\log K}{K}$, the total complexity goes to $O(C^K \log C \cdot N)$. $\square$

The other part, the cost of computing the likelihood value for each assignment, is affected by the number of mappings observed. In our case, where one peer does not map to a large number of other peers, we have the following theorem:

THEOREM 5.2. *The complexity of the one-shot estimator for each EM iteration is $O(BKC^K N \log C)$, where B is the maximal number of acquaintances a peer in the PDMS has.*

PROOF. When searching in a segment of length K, the number of observed mapping quality for each segment of length K is bounded by $O(BK)$, therefore, computing the likelihood for an assignment has the cost of $O(BK)$. With lemma 5.1, the total complexity of the one-shot estimator is $O(BKC^K \log C \cdot N)$. $\square$

Note that while there is no theoretical guarantee on the number of iterations after which the EM process will converge, in our empirical study (Section 6), we observed that most runs converge within a small number of iterations.

5.5.6 Reordering peers for segmented local search

It is easy to observe from the segmented local search procedure that the assignments carried over for the first several segments are very important. If these assignments are inferred correctly, better assignments will be discovered for later segments. As the quality of the inference is largely affected by the amount of available information (i.e., the quality of existing mappings) when processing these segments, the peers can be reordered so that as much information as possible can be used in the early stages of segmented local search. Figure 8 shows an example of this reordering for 20 peers, numbered 1 to 20. The established mappings are shown as X's. The X's in corresponding boxes are what the segmented local search can use for segments s_1 to s_4.

We use a fast greedy algorithm to re-order peers as follows. Starting with an ordered set, S, having one peer with the maximal number of established mappings in the PDMS, we add in each step a peer that maximizes the total number of observed mappings among peers in S. This process is repeated until all peers are added into S. This re-ordering can be done in $O(N^2)$ time using an efficient proper data structure. Figure 8(b) shows that more information can be used by the local search in early segments.

5.5.7 Initialization strategy

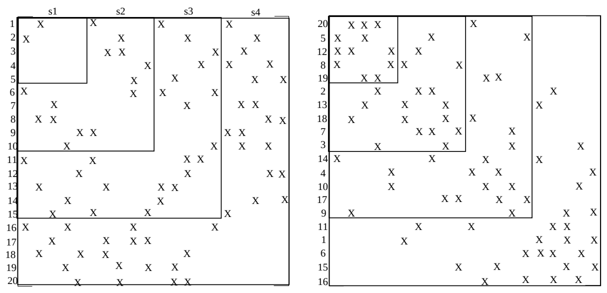

(a) before reordering (b) after reordering

Figure 8: An example of reordering 20 peers

The one-shot estimator starts with an initial θ parameter for the Gaussian distribution (Section 5.5.1). A carefully initialized θ matrix can bring extra benefit to the estimator from the following two aspects.

We observe that if we use **neutral** clusters, we may further reduce the search space for the first segment without any quality loss. Here neutral means the $|C|$ clusters are initialized with same parameters which means the "name" of the cluster does not affect the cluster assignment in our problem. A θ matrix corresponding to such a setting can be written as

$$\theta_{|C|\times|C|} = c_1 I_{|C|} + c_2(1 - I_{|C|})$$

where I is the identity matrix and c_1, c_2 are two (μ, δ^2) pairs with $c_1.\mu > c_2.\mu$. For example,

$$\begin{pmatrix} (0.7, 0.02) & (0.3, 0.04) & (0.3, 0.04) \\ (0.3, 0.04) & (0.7, 0.02) & (0.3, 0.04) \\ (0.3, 0.04) & (0.3, 0.04) & (0.7, 0.02) \end{pmatrix}$$

is a "neutral" cluster setting where $c_1(\mu, \delta^2) = (0.7, 0.02)$ and $c_2 = (0.3, 0.04)$.

If θ is initialized this way, then we can see that given an assignment, changing the label of clusters will not affect the likelihood value. Thus, we can avoid testing this kind of **equivalent assignments**. The equivalence relation can be defined as follows.

Definition 6. *[Equivalent Assignments] Two assignments Z_1 and Z_2 are considered as equivalent under parameter θ if the following conditions are satisfied.*

1. *There exists a mapping $P(x) : C \mapsto C$ that $\forall i, P(Z_1[i]) = Z_2[i]$*
2. *For all $P(x) = y$, $\theta[x][k] = \theta[y][k] \forall k = 1..|C|$*

□

To have an idea of how much we can save if we can avoid testing equivalent assignments, for $|C| = 3$ and $K = 5$, in the $3^5 = 243$ assignments the number of non-equivalent assignments is only 41, an approximately $1/C!$ savings. [2] When testing assignments, we can avoid traveling through those equivalent assignments by using the automata shown in Figure 9.

[2]The accurate number of non-equivalent assignments can be obtained by calculating $T = \sum_{i=1}^{|C|} P(i)$, where $P(i) = [i^K - \sum_{j=1}^{i-1} \binom{i}{j} P(j)]/i!$ and $P(1) = 1$

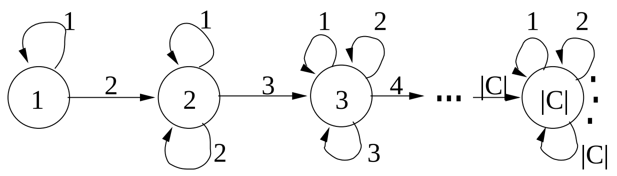


Figure 9: The automata for generating non-equivalent assignment series.

The one-shot selection scheme relies on the one-shot estimator to estimate the direct mapping quality between the host peer and a candidate acquaintance. It uses the max-min estimator (Section 5.4) to estimate the current aggregate mapping quality for the candidate and computes the selection criteria as described in Section 5.2. All candidates are then ordered on these criteria values, which represents the benefit of mapping the host peer to the candidate. Again we point out that the host peer will not always choose its new acquaintance from within its own cluster; both estimates affect the final decision.

To summarize, the one-shot acquaintance selection scheme manipulates existing pairwise mapping quality information to cluster peers and further infer the unobserved mapping quality which, combined with the current mapping quality estimation, guides the acquaintance selection. One concern is how much such mapping quality needs to be observed so that the one-shot estimator can perform accurately, especially when the number of peers increases. Our empirical study in Section 6 shows that when the number of peers increases, the required number of existing observations does not go up quickly, which makes the selection scheme practical. Our theoretical analysis showed that the estimation algorithm scales well with PDMS size; this is also validated in our empirical study.

The fact that one-shot estimator requires prior knowledge on the number of clusters makes this scheme inapplicable for scenarios in which this information is unavailable. For acquaintance selection under such scenarios, we have developed another selection scheme that does not rely on this prior knowledge: the two-hop selection scheme.

5.6 The two-hop selection scheme

In section describes a novel two-hop acquaintance selection scheme. The two-hop selection scheme uses a new two-hop direct mapping estimator, which differs than the previous one-shot scheme in the following aspects:

1. The two-hop estimator does not need to know the number of clusters that peers potentially form. Thus the scheme is applicable when that information is unavailable.
2. Under the two-hop scheme, a peer explores the network in multiple rounds, requiring fewer messages to be transmitted during each probe step. However, it does not use all established mapping information in the PDMS, which means that accuracy may be lower. We show in Section 6 that two-hop is still quite accurate.
3. New peers joining the network and existing peers leaving the network do not affect other peers' selection procedures, which makes the two-hop selection scheme suitable for PDMSs which witness frequent peer updates.
4. In additional to using established mapping quality in the PDMS, the two-hop scheme can utilize other peers' direct

mapping quality estimations for estimating direct mapping quality. With heuristics (Section 5.6.3 enabled, it utilizes the current aggregate mapping quality information.

Before detailing the operation layer steps, we first describe the new direct mapping quality estimator, namely the "two-hop estimator". Instead of collecting the pairwise mapping quality as in the one-shot estimator, the two-hop estimator focuses on mapping paths of length 2, thus receiving the name "two-hop".

5.6.1 The two-hop estimator in detail

We start with the definition of a two-hop path.

Definition 7 (two-hop path). *A path (i,k,j) is a two-hop path from i to j if and only if the mapping quality or mapping quality estimation of both mappings (i,k) and (k,j), for some other peer k, are known to peer i.* □

To estimate the direct mapping quality between peer i and peer j, the two-hop estimator computes all the two-hop paths from i to j. It uses this information to estimate the direct mapping quality $S(M_{i\_j})$, where S is the quality measure defined in Definition 1.

Let H denote the set of two-hop paths. For each two-hop path $p(i,k,j) \in H$, the two-hop estimator first estimates the direct mapping quality on path p, denoted as $S_p(M_{i\_j})$ using its expectation. Formally:

$$\begin{aligned} S_p^{est}(M_{i\_j}) &= E(S_p(M_{i\_j})) \\ &= \int_0^1 tP(t|S(M_{i\_k}),S(M_{k\_j}))dt \end{aligned} \tag{1}$$

where $P(t|S(M_{i\_k}),S(M_{k\_j}))$ is computed using a knowledge base (KB) which is a pseudo count array where each element $KB[a][b][c]$ represents the number of instances observed for a two-hop path (i,k,j) with $S(M_{i\_k})=a$, $S(M_{k\_j})=b$ and $S(M_{i\_j})=c$. Using KB, this conditional probability P can be approximated using the sample probability P^* by

$$\begin{aligned} &P(t|S(M_{i\_k}),S(M_{k\_j})) = P^*(t|S(M_{i\_k}),S(M_{k\_j})) \\ &= \frac{KB[S(M_{i\_k})][S(M_{k\_j})][t]}{\int_0^1 KB[S(M_{i\_k})][S(M_{k\_j})][t]dt} \end{aligned} \tag{2}$$

In the implementation of the two-hop estimator, the mapping quality metric (the metric in Definition 1 has range $[0,1]$), and is partitioned into T equal width intervals. Elements in KB are indexed by integers $i \in [0,\ldots,T-1]$. An entry $KB[i]$ on one of its dimension, covers mapping quality valued in interval $[i/T,(i+1)/T)$. Therefore, Equation 2 is re-written into

$$P(t|S(M_{i\_k}),S(M_{k\_j})) = \frac{KB[a][b][c]}{\sum_{u=0}^{T-1} KB[a][b][u]}$$

where $a=\lfloor T\cdot M_{i\_k}\rfloor$,$b=\lfloor T\cdot M_{k\_j}\rfloor$ and $c=\lfloor T\cdot t\rfloor$.

The two-hop estimator aggregates the estimates from the mapping quality on all two-hop paths in H together and return the final estimation of direct mapping quality: $S^{est}(M_{i\_j})$. A simple aggregation[3] takes the mean

$$S^{est}(M_{i\_j}) = \frac{1}{|H|}\sum_{p\in H} S_p^{est}(M_{i\_j}) \tag{3}$$

[3]An improved aggregation is discussed in Section 5.6.3

where $S_p^{est}(M_{i\_j})$ is as computed in Equation 1.

We now look at the two-hop selection scheme in detail.

5.6.2 The operation layer in two-hop

Peers using the two-hop selection scheme need multiple rounds discover all peers in the PDMS. The host peer discovers new peers in the probe step for each round. Each probed peer returns three types of information to the host peer: I. the existence of some other peers, II. mapping quality of established mappings, III. (updated) direct mapping quality estimation. The host peer uses the type II and III information obtained in this step to update the knowledge base. The newly discovered peers are considered as acquaintance candidates. After the host peer has finished the probe phase, it goes into the evaluation phase. First the host peer re-evaluates the direct mapping quality between itself and all peers in its candidate list — except those which were just added in the probe phase using the two-hop estimator . After this process finishes and all mapping quality estimation between itself and those candidates is updated, the host peer estimates the directly mapping quality for newly added peers.

The reason for carrying out two rounds of two-hop estimation is as follows. We believe that the information collected from most recent probe phase helps to obtain more accurate two-hop estimation. The mapping quality estimation between host peer and peers that are not new to it will be used in the estimation of the host peer and the newly probed peers. Therefore, a re-estimate of mapping quality will likely to improve the estimation quality for the new peers in the first place.

In the evaluation phase, $S(CM)$, the current aggregate mapping quality (Definition 3) is computed using the technique in Section 5.4. The difference between this and that in the one-shot scheme is that the $S(CM)$ is only computed for the set of peers discovered to the host peer instead of all peers in the network. In other words, in the one-shot scheme, the topology w.r.t. a host peer will always be the whole network while in two-hop it grows along with the probe operations.

After both $S(M)$ and $S(CM)$ are computed, the host peer picks its acquaintances using a selection criteria, for example the one described in Section 5.2.

Because the two-hop selection scheme disseminates and updates information in multiple rounds, the bookkeeping phase needs to remember when a peer has already received updated information so that the same message will not be sent to a peer twice, thus avoiding unnecessary communication overhead. This can be done by time-stamping the messages and also time-stamping each peer to indicate the last time it was updated. Unlike in one-shot estimator, a host peer using the two-hop scheme needs multiple steps to discover the whole network. A natural concern is whether it will take a long time for a peer to discover a peer that it should create mapping to. The following theorem guarantees that it does not take long for a peer to discover the whole network.

Theorem 5.3. *For a network which each peer has in average $K(K \geq 3)$ random outbound mappings, it takes in average $O(\log\log N)$ steps for a peer to discover all peers in the network using the two-hop scheme, where N is the number of peers in the network.*

PROOF. This result follows from results reported in [9] which states that the expected distance between two vertices in a connected network of size N is $O(\ln N)$. Additionally, the mean square deviation is small, so the actual distance will not deviate much from the expectation.

Now suppose path p is the **shortest path** from i to j and its length is $p = O(\log N)$. Because all peers conduct their "probe" operation simultaneously, this length is halved after each round of probing. So it takes $O(\log \log N)$ rounds for one peer to discover the existence of another peer — if there exists a path between these two peers. □

Before two-hop can be used on a PDMS, the knowledge base must be trained. It is trained on a distribution of cord quality given the two mappings (hops) on which the cord is defined. The training process can be carried on using two-hop selection in another PDMS with mapping quality fully observed or the entries KB can be initialized according to a prior distribution. If the training data's distribution mismatches the distribution of the running PDMS, the estimation from two-hop estimator will be inaccurate. This is generally true for all algorithms that rely on pre-trained models. However, the two-hop scheme keeps updating its knowledge base so that KB is updated using the observations reflecting the distribution of the running PDMS. In other words, it actively learns to improve the accuracy. In our empirical study in Section 6, we purposely use a knowledge base trained with data that has different distribution than the test data. In practice, if the peer has a strong prior knowledge on the PDMS, it will train KB with a big training data set so that updates from the running PDMS do not change much of the KB's implied distribution. On the other hand, training KB using a small training data set allows it to adopt to a new (observed) distribution more quickly.

Next we describe some useful heuristics that help to improve the accuracy of two-hop estimator.

5.6.3 Heuristics to improve the two-hop estimator

There are several opportunities to improve the basic two-hop estimator described in Section 5.6.1. The first is that simply taking the mean of all estimation from individual two-hop paths may not be the best aggregating strategy to obtain a good estimation of $S(M_{i\_j})$. Consider the example in Figure 10, where peer i wishes to estimate the direct mapping quality to peer j using the two-hop estimator. There are three two-hop paths to j. Path (i, a, j) contains one existing mapping and one estimation of mapping quality. Knowledge about path (i, b, j) comes from a previous estimation of $S(M_{i\_b})$ and $S(M_{b\_j})$. The path (i, c, j) is formed by two existing mappings. The quality of established mappings is always accurate, while quality estimations may contain errors. Therefore, two-hop may work better if we assign different weights so that established mappings receive higher weight in the estimation. This is our first heuristic:

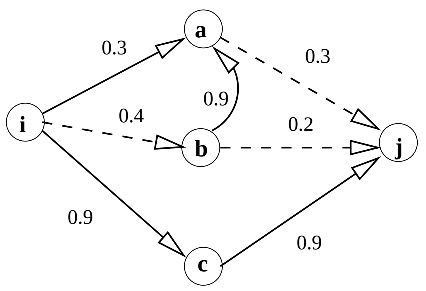


Figure 10: A motivating example for heuristics in two-hop estimation: labeled circles denote peers, solid links denote established mappings, and dotted links denote the estimation of mapping quality obtained. The number beside each link denotes the (estimated) mapping quality.

Heuristic 1: We assign different weights to paths that are formed by established mappings, estimated mapping quality, or the combination of the two. A two-hop path formed by two established mappings is given the highest weight, a path with two mapping quality estimations is given the lowest weight. The weight for a path that has one established mapping and one estimated mapping quality lies between. Let w_p denote the weight for a path, then the aggregate equation 3 is re-written as

$$S^{est}(M_{i\_j}) = \frac{\sum_{p\in H} w_p S_p^{est}(M_{i\_j})}{\sum_{p\in H} w_p}$$

□

Next, we observe that the current aggregate mapping quality gives valuable information on the quality of the direct mapping. Using the assumption that a direct mapping dominates a composed mapping (Section 5.1), the quality of the current aggregate mapping can be used as a lower bound of the direct mapping quality estimation. This results in our second heuristic for the two-hop estimator:

Heuristic 2: During acquaintance selection, the two-hop selection scheme first estimates the quality of the current aggregated mapping $S(CM_{i\_j})$ for host peer i and candidate peer j. Then the two-hop estimator uses $S(CM_{i\_j})$ as a lower bound in estimating the direct mapping quality $S(M_{i\_j})$. I.e., instead of computing the probability as in Equation 2, we compute the probability conditioned on $S(CM_{i\_j})$ as follows:

$$P(t|S(M_{i\_k}), S(M_{k\_j}), S(M_{i\_j}) \geq S(CM_{i\_j}))$$
$$= \begin{cases} \frac{KB[S(M_{i\_k})][S(M_{k\_j})][t]}{\int_u^1 KB[S(M_{i\_k})][S(M_{k\_j})][t]dt}, & t \geq S(CM_{i\_j}); \\ 0, & \text{otherwise.} \end{cases}$$

where $u = S(CM_{i\_j})$.

By replacing P in Equation 2 with this conditional probability, the mean is shifted to a more accurate value and deviation of the estimation is reduced. □

The third observation is more complicated. Still consider the example in Figure 10. In the one-shot estimator, where belief propagation is used, the inference is made based on the result of a Bayes treatment of all the information collected from the Bayes network shown in Figure 6. However, in the two-hop estimator, thus far (including the above two heuristics) the information we have considered is only a subset of information that relates to the two ending peers S and T. We have not exploited other information that may also be useful for better estimate $S(M_{S\_T})$. e.g., the correlation among peers a, b and c. For example, the fact that mapping quality from b to a is high may suggest that path (S,a,T) and (S,b,T) are correlated. When we consider quality estimation returned from these two paths, it will be beneficial that we add a penalty to account for the correlation. This gives us our next heuristic:

Heuristic 3: We give extra reward to the estimated quality from peer S to T if there are R non-correlated two-hop paths where both links' quality is estimated to be low. Then we augment the overall estimation using the following formula,

$$\hat{S} = S^{est} + \frac{H - S^{est}}{C - 1} * R$$

where $\hat{S}$ is the augmented estimation. H is a parameter larger than S^{est} and C is a number larger than or equal to 2. One observation can be helpful; if we use $C = 2$, $R = 1$ in the formula, the estimation is lifted from S^{est} to H which expresses the idea that if we believe peers form 2 clusters (C) then observing a conflicting relation(both links' quality is low) tells that the host peer and the target peer will have high(H) expected mapping quality. The problem of choosing H ,C and also efficiently determine R is out of the scope of this paper.

Similar to the one-shot scheme, the two-hop estimator and the max-min estimator work together to compute the selection criteria and rank the candidates for picking acquaintances in two-hop scheme.

5.6.4 Choosing between one-shot and two-hop

The use of different direct mapping quality estimators makes the two selection schemes suitable for different kinds of PDMSs — depending on whether the host peer knows the number of potential clusters in the PDMS. When we have no such information, we can only use the two-hop scheme. Although the two-hop selection scheme can be applied virtually to any PDMS, deciding which of the one-shot or two-hop scheme to be use is non-trivial. As we show in Section 6, the one-shot estimator — which uses all established mappings in the PDMS to do inference — does not need a large number of mappings to be observed before it can perform well. Additionally a highly mismatched KB can mislead the two-hop estimator until updates from the current PDMS change the distribution in KB to one that positively helps estimation. Therefore, for such scenarios, the one-shot selection scheme is more preferred. While the estimation from the one-shot estimator is based on all established mapping quality in the PDMS and is potentially more accurate, the two-hop selection scheme has distinct advantages on aspects we have mentioned at the beginning of Section 5.6, and as we show in the next section, performs very well in practice.

6. EMPIRICAL STUDY

In this section we present our empirical evaluation of the two acquaintance selection schemes in the previous sections. In particular, the following components are tested:

1. The max-min estimator (CP-CM) used by both the one-shot and two-hop acquaintance selection schemes. (Sections 5.3 and 5.4)
2. The one-shot estimator (OSME) and one-shot selection scheme (Section 5.5).
3. The two-hop estimator (THME) with heuristics and two-hop selection scheme (Section 5.6).

All algorithms are implemented in C++. The simulation platform is a PC running Windows XP with an Intel core2 duo 2.4G CPU and 1 Gigabyte of RAM.

Table 1 lists the meanings of notations used in our tests.

We use synthetic data sets in our experiments. The topology data is a set of randomly generated D-regular connected

Notation	Meaning
N	number of peers in PDMS
D	Initial Connectivity of D-regular graph
C	Cluster number for one-shot theme
Error	RMSE [a] for a set of estimations

[a]Root Mean Square Error: defined as $\sqrt{\frac{1}{N}\sum_{i=0}^{N}(\hat{e}-e)^2}$, where $\hat{e}$ is the estimation and e is the true value

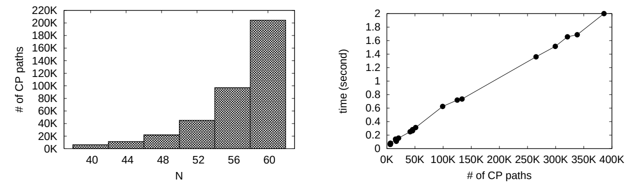
Table 1: Notations in empirical study

graphs which are used to represent the initial established mappings among peers in the PDMS. A graph is represented as an adjacency matrix A where $A_{ij} = 1$ means the mapping between peer i and peer j is established before a selection scheme starts. The D-regular property ensures that all peers have the same number of initial acquaintances; thus they all have a similar chance to discover good new acquaintances. While in practice a PDMS may have another initial topology, if peers act independently and wish to select a similar number of acquaintances, then the network topology will gradually migrate to "D-regular" for some value D. Therefore our topology setting is representative. The mapping and mapping quality data we use are randomly generated with parameters hidden to the testing schemes, except that the potential cluster number is known to the one-shot estimator. We also acknowledge that because we simulate all selection process using a single PC, networking factors are ignored. However, our empirical study suffices to test the quality of the selection scheme and the extra computation overhead that the two selection schemes impose.

6.1 Evaluation of cordless path finding algorithm

The key factor affecting the performance of the max-min estimator (Section 5.4) is the performance of the cordless path finding algorithm described in Section 5.3. We first performed a test on the performance of the cordless path finding algorithm.

Figure 11(a) shows the number of cordless paths (CP) between a randomly chosen peer and **all other** peers in a PDMS with a 20-regular graph as the initial topology. We varied the network size from 40 to 60 peers. Figure 11(b) shows the time for the cordless path finding algorithm to compute all the paths, along with the max-min estimates.



(a) #CP vs. N, D=20 (b) time vs. #CP, D=20

Figure 11: Evaluation of the CP finding algorithm

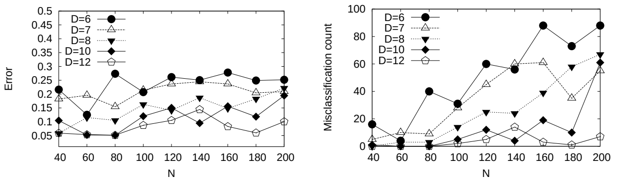
Figure 11(a) shows that when the PDMS size, N, gets large, the number of cordless paths increases very quickly for the tested $D = 20$. This suggests that indeed we need a fast estimation of the current aggregate mapping quality. Figure 11(b) validates our claim in Theorem 5.1 that the running time of the cordless path finding algorithm is linear in the number of cordless paths. The results also suggest that the algorithm performs very quickly: it computes around $400K$ paths in 2 seconds. The main reason is that the CP finding algorithm discovers new paths by extending already discovered paths so that even long paths can also be discovered very quickly. Since the cordless path algorithm performs quickly, the overhead of max-min estimator is small.

6.2 Evaluation of the one-shot selection scheme

Next we conducted a set of experiments on the one-shot selection scheme. We first tested the accuracy of the one-shot estimator (OSME). We set the initial number of acquaintances to vary from $D = 6$ to 12 and kept the cluster number fixed to $C = 4$. We let the number of peers in the PDMS vary from $N = 40$ to 200. The true in-cluster mapping quality distribution was set to $N(\mu, \delta^2) = N(0.8, 0.02)$ and the inter-cluster mapping quality followed $N(0.3, 0.05)$. The distribution parameters were hidden to the one-shot estimator. The one-shot estimator initialized with $N(0.7, 0.05)$ for in-cluster quality distribution and $N(0.4, 0.05)$ for inter-cluster quality distribution. i.e., the initial θ is

$$\begin{pmatrix} (0.7, 0.05) & (0.4, 0.05) & (0.4, 0.05) & (0.4, 0.05) \\ (0.4, 0.05) & (0.7, 0.05) & (0.4, 0.05) & (0.4, 0.05) \\ (0.4, 0.05) & (0.4, 0.05) & (0.7, 0.05) & (0.4, 0.05) \\ (0.4, 0.05) & (0.4, 0.05) & (0.4, 0.05) & (0.7, 0.05) \end{pmatrix}$$

This experiment tested whether the one-shot estimator is able to accurately infer the true cluster setting.

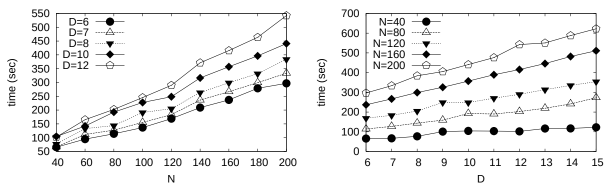


(a) Estimation error (b) Misclassification count

Figure 12: Estimation error of OSME

Figure 12 shows the RMSE error of the one-shot estimator under different settings. The results show that the one-shot estimator in general is more accurate when more initial mappings are observed (i.e., D is larger). Both figures show this trend. In general, a lower misclassification count leads to a smaller estimation error. Note that for $D = 10$ and 12, the misclassification count is 0 for $N = 80$, as shown in Figure 12(b), but the corresponding estimation error is above zero. This error comes from OSME's error in inferring the parameter (θ). From the results we can see that the error on θ is also small (about 0.05 in RMSE).

We also observed the running time of the one-shot estimator. The timing results use the same settings as for testing accuracy, and are shown in Figure 13. Each figure shows the time for **each** EM iteration in the estimation. We did not show the total time because the EM processes require different iterations to converge in different runs. Therefore, the total time does not show a trend when N or D varies, although we observed in our experiments that most runs converge within 3 iterations. The results presented in Figure 13(a) show that the running time for each iteration increases linearly with the size of the PDMS, which matches



(a) OSME time vs. N (b) OSME time vs. D

Figure 13: Time consumption of OSME

our theoretical analysis in Theorem 5.2 (Section 5.5.5). Figure 13(b) shows that the time consumption on each iteration also increases linearly with the increment of average initial observation amount (D). Our setting of using D-regular graph has lead the experiment results to perfectly validate the claim we have made in theorem 5.2 that the time complexity for each iteration is linear with the upper-bound (which is D) of each peer's acquaintance number.

Finally, we empirically showed how many established mappings are required for the one-shot estimator to make good estimations. The results are shown in Figure 14. For a PDMS with less than 80 peers, letting each peer map to 7 other peers provides enough information for the one-shot estimator to make fairly accurate estimations. For larger PDMSs (up to 200 peers), increasing the mapping connectivity to 11 provides adequate information for the one-shot estimator. One such result is shown in Figure 14, and we observed similar behavior in other experiments.

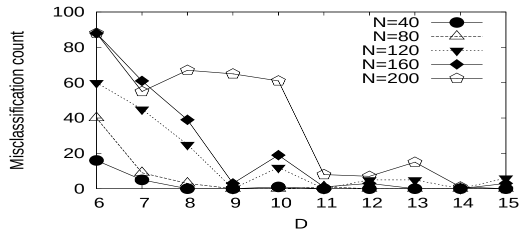


Figure 14: Requirements on initial information

From the tests in this section, we can conclude that the one-shot estimator shows a good estimation accuracy with a small amount of mapping quality information. Together with the results for the max-min estimator, we conclude that the one-shot selection scheme scales well with PDMS size.

6.3 Evaluation of the two-hop selection scheme

In this section, we evaluate the two-hop acquaintance selection scheme. First we conduct a set of experiments on the speed for a peer to explore the PDMS to validate our claim in theorem 5.3. We compute the maximal shortest distance(MSD) between any two peers in a PDMS with different random D-regular mapping topology. Let this distance be d, then a peer will take $O(\log d)$ rounds to discover the existence of all other peers in the PDMS if they also carry out two-hop selection. And if the host peer is the only one in the PDMS to perform the two-hop selection, then it needs $O(d)$ rounds to discover the entire population. The results in Figure 15 show the d value under different settings of PDMSs.

The results show that when each peer has more than 5

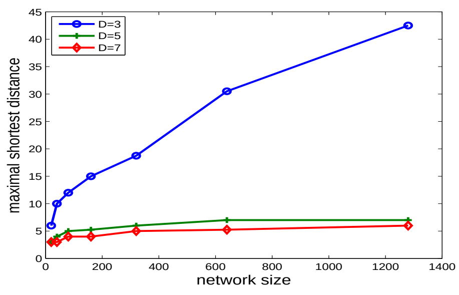


Figure 15: Maximal pairwise shortest distance in random graphs with different connectivity

acquaintances, a peer can discover other peers very quickly. This validates the claim in Theorem 5.3.

We next evaluated the accuracy of the two-hop acquaintance selection on 3 sizes of PDMSs. In all sizes, the initial topology was a 12-regular random graph. All peers in the PDMS carry out two-hop acquaintance selection in a round-robin fashion for 9 rounds. We let each peer choose the top 3 candidates as its acquaintances for each round. The knowledge base we use in the test is learned from a small training PDMS having 10 peers and with different mapping quality distribution than that used to test the PDMS. This setting was purposely chosen to simulate that in the real world, peers don't know the distribution of two-hop path quality and can only blindly start with a KB and hope to refine the knowledge base using information obtained from the running PDMS.

First we examined how well the peers selected their 3 acquaintances in each round. We evaluated this as follows. For each of the 3 acquaintances chosen by a peer in each round, we checked with an "oracle" (which knows the true mapping quality and computes exact criteria value for each candidate) to see if the peer chose a candidate from the best $t\%$ of the available candidates. A selection is considered as a "hit" if its rank is in the best $t\%$. We computed the hit ratio as the "hit" number over 3. We started with $t = 20$ and tightened the measure by decreasing t. The results in Figure 16 show the average hit rate for **all** peers in the PDMS at each round, for different sizes of PDMSs tested. We show the results for $t = 6$ because it is the first time we can observe some significant misses of the best $t\%$. The results suggest the acquaintances selected by the two-hop selection scheme are very good. Thus, although the estimation from the one-shot estimator is based on all established mapping quality in the PDMS and potentially higher, two-hop still performs quite well.

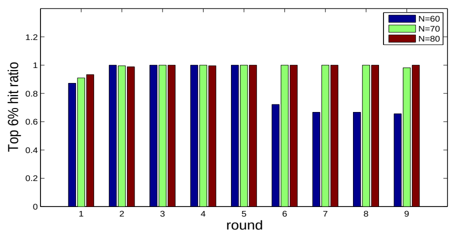


Figure 16: Hit ratio when $t\%$ tightens to 6%

We further measured the significance of the two-hop acquaintance selection qualitatively by comparing the actual mapping quality benefit for the selected acquaintances against the average benefit when a peer chooses its acquaintances randomly from its candidates. The benefit is computed using the selection criteria in Section 5.2. The results for a network with 50 peers are presented in Figure 17 where we use a box plot [29] to visualize several key statistical characteristics, detailed as follows.

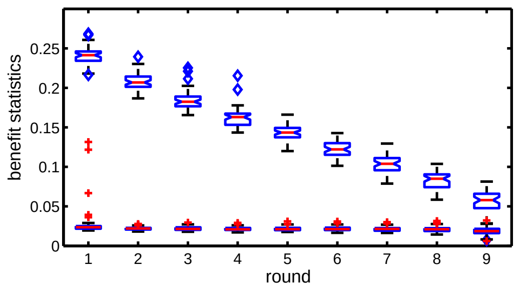
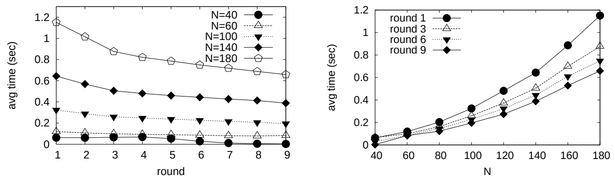


Figure 17: two-hop selection vs. random selection

In Figure 17, the notched boxes represent the gain achieved when all peers to map to their selected acquaintances in each round ($50 \times 3 = 150$ selections per round). The lower series of boxes are the corresponding results for a random selection for each round. The result shows the gain achieved by the two-hop selection scheme outperforms that of a random selection, especially for the first several rounds. The descending trend of the benefit also suggests that peers capture best acquaintances at early stages of two-hop selection. Therefore, we conclude from the previous experiments that the two-hop selection scheme can greatly improve peers' query answering potential in a PDMS.

We also tested the time consumption of the two-hop selection scheme. In this set of experiments, we let the size(N) of the PDMS vary from 40 to 180 peers. Each PDMS started with a 7-regular initial topology and a peer selected 3 acquaintances each round. The results are shown in Figure 18. Figure 18(a) compares the average time (over all selections) in selecting one acquaintance in different rounds using the two-hop selection scheme. Figure 18(b) shows how this time changes over different sizes of PDMSs.



(a) avg. time vs. round (b) avg. time vs. N

Figure 18: Time consumption of THME

The results in Figure 18(a) show that peers choose their acquaintances very quickly. Although the time needed for each selection increases when N becomes large, selecting one acquaintance requires less than 1.2 seconds. In a single run, the time needed for later rounds is smaller than that for early rounds. This is because the number of cordless paths is large for the initial topology ($D = 7$); as more acquaintances are selected, this number drops, thus saving time in computing current aggregate mapping quality. Figure 18 gives a clear view of how the two-hop scheme scales with the PDMS size in practice: time increases a little faster than linearly in N.

In summary, we tested both the accuracy and the time overhead of the two selection schemes. The experimental results show that both schemes help peers in a PDMS choose good acquaintances. Time overhead is low, and both schemes scale well in the size of the PDMS. We did not compare the two schemes because they work under different assumptions and suit different PDMS scenarios.

7. CONCLUSION AND FUTURE WORK

In this paper, we proposed two acquaintance selection schemes to help peers in a PDMS choose acquaintances that best enhance their query answering abilities. Both schemes are analyzed in theory and evaluated empirically. Our empirical evaluation shows that both the one-shot and two-hop schemes work very efficiently and help peers to choose their best acquaintances effectively. Moreover, our theoretical analysis and empirical study both show that the two schemes scales to PDMSs with a large number of peers.

There are a number of extensions we wish to further explore in the future. We will continue to explore the possibility that peers may co-operate together in acquaintance selection procedure so to achieve an even better query answering ability. We also wish to study the peers' behavior using game theory; it may lead to interesting results.

Acknowledgement

We thank David Kirkpatrick for valuable discussion on the cordless path algorithm and Lin Xu for helpful discussion.

8. REFERENCES

[1] Sameer Agarwal, Jongwoo Lim, Lihi Zelnik-Manor, Pietro Perona, David J. Kriegman, and Serge Belongie. Beyond pairwise clustering. In *CVPR(2)*, 2005.

[2] Philip Bernstein, Fausto Giunchiglia, Anastasios Kementsietsidis, John Mylopoulos, Luciano Serafini, and Ilya Zaihrayeu. Data management for peer-to-peer computing: A vision. In *WebDB*, pages 89–94, 2002.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] Miguel Castro, Peter Druschel, Y. Charlie Hu, and Antony Rowstron. Proximity neighbor selection in tree based structured peer-to-peer overlays. Technical report, Microsoft, 2003.

[5] Vicent Cholvi, Pascal Felber, and Ernst W. Biersack:. Efficient search in unstructured peer-to-peer networks. In *SPAA*, 2004.

[6] Byung-Gon Chun, Ben Y. Zhao, and John Kubiatowicz:. Impact of neighbor selection on performance and resilience of structured p2p networks. In *IPTPS*, 2005.

[7] Curt Cramer and Thomas Fuhrmann:. Proximity neighbor selection for a dht in wireless multi-hop networks. In *Peer-to-Peer Computing*, 2005.

[8] AnHai Doan and Alon Y. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26(1):83–94, 2005.

[9] S.N. Dorogovtsev, J.F.F. Mendes, and A.N. Samukhin. Metric structure of random networks. *Nucl. Phys. B 653, 307*, 2003.

[10] Shlomo Dubnov, Ran El-Yaniv, Yoram Gdalyahu, Elad Schneidman, Naftali Tishby, and Golan Yona. A new nonparametric pairwise clustering algorithm based on iterative estimation of distance profiles. *Machine Learning*, 47(1):35–61, 2002.

[11] Ronald Fagin, Phokion G. Kolatis, Lucian Popa, and Wang Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. In *PODS*, pages 83–94, 2004.

[12] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Piazza: Data management infrastructure for semantic web applications. In *ICDE*, pages 505–516, 2003.

[13] Iraklis A. Klampanos and Joemon M. Jose. An architecture for information retrieval over semi-collaborating peer-to-peer networks. In *SAC*, 2004.

[14] Alexander Loser, Felix Naumann, Wolf Siberski, Wolfgang Nejdl, and Uwe Thaden. Semantic overlay clusters within super-peer networks. In *DBISP2P*, pages 33–47, 2003.

[15] Jayant Madhavan and Alon Y. Halevy. Composing mappings among data sources. In *VLDB 2003*, 2003.

[16] Federica Mandreoli, Riccardo Martoglia, Simona Sassatelli, Paolo Tiberio, and Wilma Penzo. Using semantic mappings for query routing in a pdms environment. In *SEBD*, 2006.

[17] Cheuk Hang Ng, Ka Cheung Sia, and Chi-Hang Chan. Advanced peer clustering and firework query model in the peer-to-peer network. In *WWW(Poster)*, 2003.

[18] Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(1):167–172, 2007.

[19] Murali Krishna Ramanathan, Vana Kalogeraki, and Jim Pruyne. Finding good peers in peer-to-peer networks. In *IPDPS*, 2002.

[20] Antonio Robles-Kelly and Edwin R. Hancock. Pairwise clustering with matrix factorisation and the em algorithm. In *Conference on Computer Vision*, pages 63–77, London, UK, 2002. Springer-Verlag.

[21] Patricia Rodríguez-Gianolli, Maddalena Garzetti, Lei Jiang, Anastasios Kementsietsidis, Iluju Kiringa, Mehedi Masud, Renée J. Miller, and John Mylopoulos. Data sharing in the hyperion peer database system. In *VLDB*, pages 1291–1294, 2005.

[22] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.

[23] Noam Shental, Assaf Zomet, Tomer Hertz, and Yair Weiss. Pairwise clustering and graphical models. In *Neural Information Processing Systems*, 2003.

[24] Kunwadee Sripanidkulchai, Bruce M. Maggs, and Hui Zhang:. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM 2003*, 2003.

[25] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek,

and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003.

[26] Igor Tatarinov and Alon Y. Halevy. Efficient query reformulation in peer data management systems. In *SIGMOD*, pages 539–550, 2004.

[27] Christoph Tempich, Alexander Loser, and Jorg Heizmann:. Community based ranking in peer-to-peer networks. In *ODBASE*, 2005.

[28] Dimitrios Tsoumakos and Nick Roussopoulos:. Agno: An adaptive group communication scheme for unstructured p2p networks. In *Euro-Par*, 2005.

[29] John Wilder Tukey. *Exploratory Data Analysis.* Addison-Wesley Publishing, 1977.

[30] Zhe Yang Xi Tong, Dalu Zhang. Efficient content location based on interest-cluster in peer-to-peer system. In *ICEBE*, 2005.