Clustering Service Networks with Entity, Attribute and Link Heterogeneity

Yang Zhou†, Ling Liu†, Calton Pu†, Xianqiang Bao†, Kisung Lee†, Balaji Palanisamy‡, Emre Yigitoglu†, Qi Zhang†
†*Georgia Institute of Technology* ‡*University of Pittsburgh*
yzhou@gatech.edu, lingliu@cc.gatech.edu, calton.pu@cc.gatech.edu, xbao31@cc.gatech.edu
kslee@gatech.edu, bpalan@pitt.edu, eyigitoglu@gatech.edu, qzhang90@gatech.edu

***Abstract*—Many popular web service networks are content-rich in terms of heterogeneous types of entities and links, associated with incomplete attributes. Clustering such heterogeneous service networks demands new clustering techniques that can handle two heterogeneity challenges: (1) multiple types of entities co-exist in the same service network with multiple attributes, and (2) links between entities have diverse types and carry different semantics. Existing heterogeneous graph clustering techniques tend to pick initial centroids uniformly at random, specify the number k of clusters in advance, and fix k during the clustering process. In this paper, we propose ServiceCluster, a novel heterogeneous Service network Clustering algorithm with four unique features. First, we incorporate various types of entity, attribute and link information into a unified distance measure. Second, we design a Discrete Steepest Descent method to naturally produce initial k and initial centroids simultaneously. Third, we propose a dynamic learning method to automatically adjust the link weights towards clustering convergence. Fourth, we develop an effective optimization strategy to identify new suitable k and k well-chosen centroids at each clustering iteration.**

I. Introduction

Efficient web service analysis is widely recognized as an interesting and challenging research problem, which has received heated attention recently [2]–[12]. As more and more people are engaged in service network analysis, we witness many forms of heterogeneous service networks in which entities are of different types and are interconnected through heterogeneous types of links, representing different kinds of semantic relations. Figure 1 presents a real service network from IBM knowledge base. There are two kinds of object vertices: blue "Service" vertices and grey "Provider" nodes. Each "Service" vertex may contain three types of properties: red, purple and green attribute vertices specify service's "Type", "Category" and "Capability", respectively. Each "Provider" vertex may have two kinds of attributes: red "Type" attribute and purple "Category" property. On the other hand, there are three kinds of links: an ochre edge represents the "Provides" relationship between "Service"s and "Provider"s; a black line specifies the structure relationship between objects with the same type; a dashed edge denotes the attribute edge between object and its attribute.

Analyzing and mining such heterogeneous service networks can provide new insights about how entities influence and interact with each other and how ideas and opinions propagate on service networks. For example, clustering

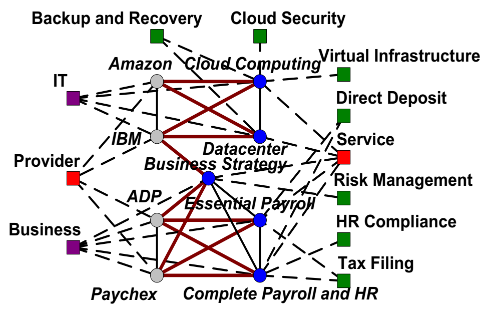


Figure 1. A Heterogeneous Service Network from IBM Knowledge Base

online service network may help understanding consumer segmentation for service marketing. Clustering heterogeneous social network becomes an interesting and challenging research problem which has received much attention recently [13], [16], [17], [19]–[22]. However, clustering heterogeneous networks with multiple types of entities, attributes and links poses a number of new challenges.

- Different types of entities co-exist in the same information network, and each type of entities may have diverse types of links and different sets of attributes. It is very challenging to decide the importance of various types of entities, attributes and links to improve the clustering quality. In Figure 1, if we want to partition "Provider"s into clusters based on the structure links between "Provider"s and their attribute information, then two associated attributes of "Type" and "Category" may have different degrees of importance. A weight learning method for different types of links in terms of their contribution in the clustering is a possible solution.
- Most of existing heterogeneous graph clustering methods [13], [16], [17], [19]–[22] require the number of clusters to be specified in advance. It is hard to decide for inexperienced users. The usual method is to compare the results of multiple runs with different k values and choose the best one in terms of a given criterion. However, the repeated clustering run could be expensive for large datasets.
- The choice of initial cluster centroids have a great effect on the clustering result. Existing studies usually choose centroids uniformly at random from data points, which makes it difficult to find high quality clustering results.
- None of existing literatures [16], [17], [19], [20], [22] on weight learning methods has studied the impact of dynamic weight assignment on the dataset. For example, the original similarity (or distance) scores may need to be updated due to iterative update of link weights. In addition, it is necessary to recalculate the better k and the better centroids

with the update on similarity (or distance) scores, which actually change the shape and scale of dataset.

With these new challenges in mind, in this paper we develop an innovative dynamic clustering approach of heterogeneous service networks, called SERVICECLUSTER. Our approach makes a number of original contributions.

- We propose a unified random walk distance measure integrating various types of entities, attributes and links to measure vertex closeness on a heterogeneous network.
- We design a Discrete Steepest Descent method to naturally determine the number of clusters and generate the corresponding centroids in a heterogeneous network.
- We propose a dynamic learning method to automatically adjust the link weights towards the direction of clustering convergence.
- We argue that the correct choice of k often depends on the shape and scale of dataset. Thus we develop an effective optimization strategy to identify new suitable k and k well-chosen centroids upon the update on similarity scores and weight tuning to continuously improve the clustering quality at each clustering iteration.
- We perform extensive evaluation of ServiceCluster on three real datasets and our experimental results show that ServiceCluster outperforms existing representative methods in terms of both effectiveness and efficiency.

II. RELATED WORK

Web service discovery and management has been a heated topic in recent years [2]–[7]. Skoutas et al. [8] proposed a methodology for ranking and clustering the relevant web services based on the notion of dominance, which apply multiple matching criteria without aggregating the match scores of individual service parameters. Xiao et al. [9] proposed a context modeling approach which can dynamically handle various context types and values. Almulla et al. [10] presented a web services selection model based on fuzzy logic and proposed a fuzzy ranking algorithm based on the dependencies between proposed quality attributes. Liu et al. [11] proposed a heuristic social context-Aware trust network discovery algorithm, H-SCAN, by adopting the K-Best-First Search (KBFS) method and some optimization strategies. Kumara et al. [12] proposed a hybrid web-service clustering approach with considering both ontology learning and IR-based term similarity.

Graph clustering and graph classification has attracted active research in the last decade [13]–[23]. Shiga et al. [13] presented a clustering method which integrates numerical vectors with modularity into a spectral relaxation problem. SCAN [14] is a structural clustering algorithm to detect clusters, hubs and outliers in networks. SA-Cluster [16], Inc-Cluster [17] and BAGC [19] perform clustering based on both structural and attribute information by incorporating attributes into an attributed graph. RankClass [18] groups objects into pre-specified classes, while generating the ranking information for each type of object. GenClus [20] proposed a model-based clustering method for heterogeneous graphs with different link types and attribute types.

Some recent studies have shown the clustering quality can be enhanced by selecting a good k or by choosing good initial centroids. K-Means++ [24] can find a clustering that is $O(logk)$-competitive to the optimal K-Means solution by specifying a procedure to initialize the cluster centers before proceeding with the K-Means iterations. G-Means [25] runs K-Means with increasing k in a hierarchical fashion until the data assigned to each K-Means center are Gaussian.

To our knowledge, this work is the first one to address the problem of clustering heterogeneous service networks by simultaneously refining the link weights, the k value and the cluster centroids to progressively enhance the clustering quality in each clustering iteration.

III. PROBLEM STATEMENT

A **heterogeneous service network** is denoted as $G = (V, A, E)$, where $V = \bigcup_{i=1}^{N} V_i$ represents the set of N types of entity vertices, such as services, providers and customers, $A = \bigcup_{i=1}^{M} A_i$ denotes the set of M kinds of associated attribute vertices, $E = \bigcup_{i=1}^{L} E_i$ specifies the set of L types of edges among entity vertices and attribute vertices. An attribute vertex $v \in A_i$ denotes some concrete value for the i^{th} kind of attributes. For example, an attribute of provider "Category" has a value of "IT". An edge may exist between entities with the same type (e.g., "Cloud Computing" and "Datacenter" are similar), or between entities with different types (e.g., "Amazon" provides a service of "Datacenter"), or between entities and attributes (e.g., "IBM"'s "Category" is "IT"). We denote the number of each type of entity vertices, attribute vertices or links as $n_i = |V_i|$ $(1 \le i \le N)$, $m_j = |A_j|$ $(1 \le j \le M)$, $l_k = |E_k|$ $(1 \le k \le L)$ respectively. The total number of entities, attributes or links is equal to $n = \sum_{i=1}^{N} n_i$, $m = \sum_{j=1}^{M} m_j$, $l = \sum_{k=1}^{L} l_k$, respectively.

Given a heterogeneous service network G, the problem of Heterogeneous **<u>Service</u> Network <u>Clustering</u>** (SERVICECLUSTER) is to partition the objective entities V_i with the i^{th} type into k_i disjoint clusters $C_1, C_2, \ldots, C_{k_i}$, where $i \in \{1, 2, \ldots, N\}$, $V_i = \bigcup_{p=1}^{k_i} C_p$ and $C_p \bigcap C_q = \phi$ for $\forall p, q, 1 \le p, q \le k_i, p \ne q$, to ensure: (1) the clustering of the objective entities has an optimal number of clusters in terms of the shape and scale of dataset; (2) the entities within each cluster are densely connected, while the entities in different clusters are distant from each other; (3) the entities within each cluster have similar interactions with other types of entities, while the entities in different clusters have diverse interplays with other types of entities; and (4) the entities within clusters may have similar properties, while the entities in different clusters have diverse attribute values.

IV. A UNIFIED WEIGHTED DISTANCE MEASURE

In a heterogeneous network with various types of entities, attributes and links, each entity is associated with a set of multiple types of entities through the links between entity vertices, and a set of different kinds of attributes through the connections between entity vertices and attribute vertices, we propose to use a unified distance measure based on the neighborhood random walk model to integrate various types of link information. In the heterogeneous network, there exists a random walk path between two entities $v_1, v_2 \in V_i$ if (1) v_1 and v_2 have the same peer entity $v_3 \in V_i$; (2) both v_1 and v_2 are connected to the same entity $v_4 \in V_j$ with different type; or (3) v_1 and v_2 have the same attribute value $v_5 \in A_k$. If there are multiple paths connecting v_1 and v_2, then they are close. On the other hand, if there are very few or no paths between v_1 and v_2, then they are far apart.

Definition 1 (Transition Probability): Let $V = \bigcup_{i=1}^{N} V_i$ be the set of N types of entities, $A = \bigcup_{i=1}^{M} A_i$ be the set of M kinds of associated attributes, the transition probability matrix T of a heterogeneous network G is defined below.

$$P = \left[\begin{array}{ccc|ccc} P_{11} & \cdots & P_{1N} & P_{1(N+1)} & \cdots & P_{1(N+M)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{N1} & \cdots & P_{NN} & P_{N(N+1)} & \cdots & P_{N(N+M)} \\ \hdashline P_{(N+1)1} & \cdots & P_{(N+1)N} & P_{(N+1)(N+1)} & \cdots & P_{(N+1)(N+M)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{(N+M)1} & \cdots & P_{(N+M)N} & P_{(N+M)(N+1)} & \cdots & P_{(N+M)(N+M)} \end{array}\right]$$

$$T = \omega \cdot P \tag{1}$$

where each P_{jk} in P represents the transition probability for some kind of links. Each ω_{jk} in weight matrix $\omega = [\omega_{11}, \ldots, \omega_{1(N+M)}; \ldots; \omega_{(N+M)1}, \ldots, \omega_{(N+M)(N+M)}]$ specifies the weight of P_{jk}. According to the types of sources and destinations, P is divided into four parts: (1) P_{jk} $(1 \le j, k \le N)$ is a $n_j \times n_k$ block matrix representing the transition probability between entity vertices. Each entry in P_{jk} is the original edge value between entities, e.g., the similar degree between two entity vertices with the same type of "Service", or denoting whether $v \in V_k$ is provided by $u \in V_j$ when u is a "Provider" and v is a "Service"; (2) a $n_j \times m_{k-N}$ block matrix P_{jk} $(1 \le j \le N, N+1 \le k \le N+M)$ specifies the transition probability from entities to attributes. Each element in P_{jk} has a binary value of 0 or 1 specifying whether the entity holds the attribute value, e.g., "IBM" has a "Category" of "IT"; (3) P_{jk} $(N+1 \le j \le N+M, 1 \le k \le N)$ is a $m_{j-N} \times n_k$ block matrix denoting the transition probability from attributes to entities. Each entry specifies whether the attribute value is owned by the entity, e.g., a "Category" of "IT" is possessed by "IBM"; and (4) P_{jk} $(N+1 \le j, k \le N+M)$ is a $m_{j-N} \times m_{k-N}$ block matrix with all 0s since there is no connection between attributes.

We argue that each type of links may have different degrees of contribution in the clustering process. Thus we assign an individual weight ω_{ij} for each kind of transition probabilities. Notice that ω_{ij} may not be equal to ω_{ji} since the information flow between two different types of vertices may be bidirectional. In our current experiment setup, all weights are initialized as 1.0. Since each row of transition probability matrix should sum to 1, we further perform the row-wise normalization for T.

Based on the definition of transition probability, we actually split the original transition operation into two steps: (1) inspect the weight of source's neighbors and choose some kind of vertices with the largest weight; and (2) check the original edge value between source and vertices with the largest weight and choose some vertex with the largest edge value as destination.

Definition 2 (Unified Neighborhood Random Walk Distance): Let T be the transition probability of a heterogeneous network G, l be the length that a random walk can go, and $c \in (0, 1)$ be the restart probability, the unified random walk distance $d(u, v)$ from $u \in V$ to $v \in V$ in G is defined as follow.

$$d(u,v) = \sum_{\substack{\tau: u \rightsquigarrow v \\ length(\tau) \le l}} p(\tau) c (1-c)^{length(\tau)} \tag{2}$$

where τ is a path from u to v whose length is $length(\tau)$ with transition probability $p(\tau)$. $d(u, v)$ reflects the vertex closeness based on multiple types of link information.

The matrix form of the unified distance is given as follow.

$$R = \sum_{\gamma=1}^{l} c(1-c)^{\gamma} T^{\gamma} \tag{3}$$

V. Heterogeneous Service Network Clustering

With the unified random walk distance as an input, ServiceCluster first identifies initial k_i and initial centroids by using a Discrete Steepest Descent method. At each inner iteration, it follows the *K-Medoids* clustering method [26]: assign vertices to their closest centroids and select the most centrally located point in a cluster as new centroid. At each outer iteration, the optimal weights are generated by maximizing the clustering objective. It further splits or merges current clusters to identify new k_i and new centroids. This process is repeated until convergence.

A. Selection of k_i and Initial Centroids

We will address two main issues in the initialization step: (1) initial k_i setup and (2) cluster centroid initialization.

We argue that choosing k_i randomly without prior knowledge by existing heterogeneous clustering methods often leads to incorrect clustering results. In addition, good initial centroids are essential for the success of partitioning-based clustering algorithms. We propose a Discrete Steepest Descent method (DSD) to provide a natural way to determine the number of clusters and the initial centroids simultaneously. Intuitively, if we choose a local densest vertex, which is similar to the most peer vertices, in its neighborhood as centroids, then this will maximize the within-cluster similarity on the group, which consists of the centroid and its neighbors. To find such local densest vertices, we first define the density $D_{V_i}(v)$ of an entity $v \in V_i$ on V_i in terms of the unified distance measure below.

$$D_{V_i}(v) = \sum_{u \in V_i, u \in \epsilon-neighborhood\ of\ v, \epsilon \in \mathbb{Z}_+} d(u,v) \tag{4}$$

where $d(u, v)$ is the unified random walk distance from u to v in G. $D_{V_i}(v)$ summarizes the similarities between v

and its ϵ-*neighborhood* in V_i. According to the definition of clustering objective in Eq.(9), $D_{V_i}(v)$ is equivalent to the objective of cluster which takes v as centroid and consists of v and its ϵ-*neighborhood* in V_i.

The Steepest Descent method (SD) is an effective first-order optimization algorithm to find a local maximum (or minimum) of a function. The local maximization version of SD starts with an initial point x_0, and iteratively executes the following steps to update the current iterate x_{t-1} by a step γ_t from x_0 in the gradient direction which increases the objective $f(x)$ until moving to a critical point, i.e., $x_t = x_{t-1}$, which is hopefully the desired local maximum.

$$\begin{aligned} \gamma_t &= argmax_{\gamma \geq 0}\ f(x_{t-1} + \gamma f'(x_{t-1})) \\ x_t &= x_{t-1} + \gamma_t f'(x_{t-1}) \end{aligned} \tag{5}$$

where γ_t is the steepest step to increase $f(x)$ at the fastest rate and therefore make the biggest change.

So far most existing first order optimization methods such as Steepest Descent or second order optimization models such as Newton-Raphson have assumed a continuous differentiable search space. However, the search space in a graph is not continuous but discrete, i.e., made up of individual vertices. Thus, we propose the DSD method to find the local maxima of $D_{V_i}(v)$ in the heterogeneous network G.

$$\begin{aligned} \gamma_t &= argmax_{\gamma : v_{t-1} \rightarrow v_t, v_t \in \epsilon-neighborhood\ of\ v_{t-1}, \epsilon \in \mathbb{Z}_+} D_{V_i}(v_t) \\ & v_{t-1} \xrightarrow{\gamma_t} v_t \end{aligned} \tag{6}$$

where the gradient $D'_{V_i}(v_{t-1})$ is approximated by $D_{V_i}(v_t) - D_{V_i}(v_{t-1})$ and the steepest step is directed to the densest vertex in the ϵ-*neighborhood* of v_{t-1}. The process terminates when $D_{V_i}(v_t) \leq D_{V_i}(v_{t-1})$. This approach is typically much faster than an exhaustive search when V_i is large.

Based on the DSD method, the initialization of k_i and centroids for the objective entities V_i is presented in Algorithm 1. Each iteration in the DSD process forms a path of steepest descent from a starting vertex v_0 to a local maximum v_{t-1} of density. When the DSD procedure carries out a multi-dimensional search from an unvisited vertex v_{t-1} to a visited vertex v_t and $D_{V_i}(v_t) > D_{V_i}(v_{t-1})$, we incorporate the current path from v_0 to v_{t-1} and the previous path including v_t into a tree of steepest descent since $v_0, \ldots, v_t$ have the same local maximum. The DSD process is repeated until each vertex is assigned to one and only DSD path or tree. Thus, we present a quite natural way to generate k_i centroids (local maxima) and k_i clusters (paths or trees).

When the heterogeneous network is dense enough, or we run enough random walk propagations on the heterogeneous network, the DSD method exhibits superior performance on the selection of k_i. Assuming that each entity is at least in the ϵ-*neighborhood* of one entity vertex with local maximum of density, we can generate the following theoretical property.

Theorem 1: The lower bound of the optimal number of clusters is k_i by DSD.

Proof. Let C_1 and C_2 be two arbitrary clusters by DSD, c_1 and c_2 be the corresponding centroids respectively. We try to prove the clustering objective will be reduced if we combine

Algorithm 1 Initialization of k_i and Centroids

Input: a service network $G{=}(V, A, E)$, the objective entities V_i, the random walk distance R, a set of unvisited entities S, a set of visited entities T.
Output: k_i, centroids $c_1, ..., c_{k_i}$.

```
1: $S{=}V_i$, $T{=}\phi$ and $k_i{=}0$;
2: while $S{\neq}\phi$
3:    Choose one starting vertex $v_0$ randomly from $S$;
4:    while $D_{V_i}(v_{t-1}){<}D_{V_i}(v_t)$ by DSD
5:      if $v_t{\in}T$ and $v_t{\in}C_k(k{\in}\{1,\ldots,k_i\})$
6:         $S{=}S{-}\{v_0,\ldots,v_{t-1}\}$ and $T{=}T{+}\{v_0,\ldots,v_{t-1}\}$;
7:         $C_k{=}C_k{+}\{v_0,\ldots,v_{t-1}\}$ and goto Step 2;
8:    $S{=}S{-}\{v_0,\ldots,v_{t-1}\}$ and $T{=}T{+}\{v_0,\ldots,v_{t-1}\}$;
9:    $k_i{=}k_i{+}1$, $c_{k_i}{=}v_{t-1}$ and $C_{k_i}{=}\{v_0,\ldots,v_{t-1}\}$;
10: Return $k_i$, $c_1, ..., c_{k_i}$ and $C_1, ..., C_{k_i}$.
```

C_1 and C_2 into one cluster, i.e, decrease k_i. There are two possible cases to be discussed separately: (1) if the centroid in the new combined cluster is arbitrary one of c_1 and c_2, say c_1, in terms of the definition of clustering objective in Eq.(9), then the cluster objective on $\forall v \in C_2$ will be reduced due to $\sum_{v \in C_2} d(v, c_1) \leq \sum_{v \in C_2} d(v, c_2)$; and (2) if the centroid c in the combined cluster is neither c_1 nor c_2, then the cluster objective on both $\forall u \in C_1$ and $\forall v \in C_2$ will be reduced due to $\sum_{u \in C_1} d(u, c) \leq \sum_{u \in C_1} d(u, c_1)$ and $\sum_{v \in C_2} d(v, c) \leq \sum_{v \in C_2} d(v, c_2)$.

B. Vertex Assignment and Centroid Update

Although the DSD method can produce a good clustering when facing a dense heterogeneous network, we may need to further refine clusters when many entity vertices locate outside the ϵ-*neighborhood* of any local maximum since each DSD step adopts a local optimization strategy in the range of ϵ-*neighborhood* of current iterate v_{t-1}. For example, there is a DSD path $v_0 \rightarrow v_1 \rightarrow v_2$ where v_0 is the starting point and v_2 is the local maximum. Although v_1 may be very similar to both v_0 and v_2, the similarity between v_0 and v_2 may be quite small.

For the objective entities V_i, with k_i centroids in the t^{th} iteration, we assign each vertex $u \in V_i$ to its closest centroid $c^* = argmax_{c_j^t \in \{c_1^t, \ldots, c_{k_i}^t\}} d(u, c_j^t)$. When all vertices are assigned to some cluster, the centroid will be updated with the most centrally located vertex in each cluster. To find such a vertex, we first compute the "average point" $\overline{u}$ of a cluster C_j in terms of the unified distance measure as

$$d(\overline{u}, v) = \frac{1}{|C_j|} \sum_{w \in C_j} d(w, v), \forall v \in V \tag{7}$$

Thus $d(\overline{u}, :)$ is the average unified distance vector for cluster C_j. Then we find the new centroid c_j^{t+1} in C_j as

$$c_j^{t+1} = argmin_{v \in C_j} \| d(\overline{v}, :) - d(\overline{u}, :) \| \tag{8}$$

Therefore we find the new centroid c_j^{t+1} in the $(t+1)^{th}$ iteration whose unified random walk distance vector is the closest to the cluster average.

C. Objective Function

The objective of clustering is to maximize within-cluster similarity between centroids and member vertices.

Definition 3: [Graph Clustering Objective Function] Let $G = (V, A, E)$ be a heterogeneous network with N types of

Algorithm 2 Splitting and Merging of Clusters

```
Input: a service network $G$=$(V, A, E)$, the objective entities $V_i$, the random walk distance $R$, $k_i$ clusters $C_1, ..., C_{k_i}$.
Output: $k_i$, centroids $c_1, ..., c_{k_i}$.
1: Calculate $D_{C_j}(v)$ for each vertex $v$, $\forall v \in C_j$, in each cluster $C_j$;
2: find all local maxima $c_{j1}, ..., c_{jk_j}$ in each cluster $C_j$ by Algorithm 1;
3: Split each cluster $C_j$ into subclusters $C_{j1}, ..., C_{jk_j}$;
4: for each local maximum $c$
5:   Compute $D_{V_i}(c)$ and $D_{V_i}(v), \forall v \in \epsilon\text{-}neighborhood$ of $c$;
6:   if $D_{V_i}(v) > D_{V_i}(c)$, $v = argmax_{u \in \epsilon-neighborhood\ of\ c} D_{V_i}(u)$
7:     Merge $C_{ps}$ and $C_{qt}$ where $v \in C_{ps}$, $c \in C_{qt}$, $C_{ps} \neq C_{qt}$;
8: Update $k_i$;
9: Return $k_i$, $c_1, ..., c_{k_i}$ and $C_1, ..., C_{k_i}$.
```

entity vertices and M kinds of associated attribute vertices, $\omega_{jk}(1 \leq j, k \leq N+M)$ be the weight of each kind of links, and k_i be the number of clusters for the objective entities V_i. The goal of the heterogeneous network clustering is to find k_i partitions $\{C_p\}_{p=1}^{k_i}$ such that $V_i = \bigcup_{p=1}^{k_i} C_p$ and $C_p \bigcap C_q = \phi$ for $\forall p, q, 1 \leq p, q \leq k_i, p \neq q$, and the following objective function $O(\{C_p\}_{p=1}^{k_i}, \omega)$ is maximized.

$$O(\{C_p\}_{p=1}^{k_i}, \omega) = \sum_{p=1}^{k_i} \sum_{v \in C_p} d(v, c_p)$$
$$\omega = [\omega_{11}; \ldots; \omega_{1(N+M)}; \ldots; \omega_{(N+M)1}; \ldots; \omega_{(N+M)(N+M)}] \tag{9}$$
$$\text{s.t. } \omega_{jk} \geq 0, \sum_{j=1}^{N+M} \sum_{k=1}^{N+M} \omega_{jk} = (N+M)^2.$$

The original objective is a polynomial function of ω with non-negative coefficients. We reformulate it as follow.

$$O(\{C_p\}_{p=1}^{k_i}, \omega) = \sum_{m=1}^{n} a_m \prod_{j=1,k=1}^{N+M} (\omega_{jk})^{p_{mjk}}$$
$$a_m \geq 0, p_{mjk} \geq 0, p_{mjk} \in \mathbb{Z}, \sum_{j=1,k=1}^{N+M} p_{mjk} \leq l \tag{10}$$
$$\text{s.t. } \omega_{jk} \geq 0, \sum_{j=1}^{N+M} \sum_{k=1}^{N+M} \omega_{jk} \leq (N+M)^2.$$

where the objective consists of n within-cluster polynomial terms, a_m is the coefficient of the m^{th} term, p_{mjk} is the exponent of ω_{jk} in the m^{th} term, and l is the length that a random walk can go. Notice that we replace the constraints in Eq.(9) with those in Eq.(10) since $O(\{C_p\}_{p=1}^{k_i}, \omega)$ is monotonically increasing with non-negative ω such that it can achieve the same maximum on both constraint spaces.

For ease of presentation, we substitute ω with μ such that their components are in one-to-one correspondence.

$$O(\{C_p\}_{p=1}^{k_i}, \mu) = \sum_{m=1}^{n} a_m \prod_{j=1}^{(N+M)^2} \mu_j^{p_{mj}}, a_m \geq 0, p_{mj} \geq 0,$$
$$p_{mj} \in \mathbb{Z}, \sum_{j=1}^{(N+M)^2} p_{mj} \leq l, \mu = [\mu_1; \ldots; \mu_{(N+M)^2}] \tag{11}$$
$$\text{s.t. } \mu_j \geq 0, \sum_{j=1}^{(N+M)^2} \mu_j \leq (N+M)^2.$$

where p_{mj} is the exponent of μ_j in the m^{th} term.

D. Weight Optimization

The original optimization problem is a high-dimensional polynomial programming problem. On the other hand, the polynomial objective contains massive variables such that we can not directly solve the KKT system of polynomial

Algorithm 3 Heterogeneous Service Network Clustering

```
Input: a service network $G$=$(V, A, E)$, the objective entities $V_i$, a length limit $l$ of random walk paths, a restart probability $c$.
Output: $k_i$ clusters $C_1, ..., C_{k_i}$.
1: $\omega$=$\mu$=$\mathbf{1}$;
2: Calculate $T$ and $R$;
3: $k_i$ and initial centroids $c_1, ..., c_{k_i}$ by Algorithm 1;
4: Repeat until the weight vector $\mu$ converges:
5:   Repeat until the objective $O(\{C_p\}_{p=1}^{k_i}, \mu)$ converges:
6:     Assign each vertex $v$ to $c^* = argmax_{c_j} d(v, c_j)$;
7:     Update $c_j = argmin_{v \in C_j} \|d(\overline{v}, :) - d(\overline{u}, :)\|$;
8:   Run the SCA method to solve $(O(\{C_p\}_{p=1}^{k_i}, \mu))$ to produce $\omega$;
9:   Re-calculate $T$ and $R$ with the optimal $\omega$;
10:  Update $k_i$ and $c_1, ..., c_{k_i}$ by Algorithm 2;
11: Return $k_i$ clusters $C_1, ..., C_{k_i}$.
```

equations. It is very hard to perform function trend identification and estimation to determine the convexity of the optimization problem. Thus, there may exist no verifiable sufficient conditions for global optimality. We convert the optimization problem in Eq.(11) to the following equivalent problem by setting $\mu_j = e^{\nu_j}$.

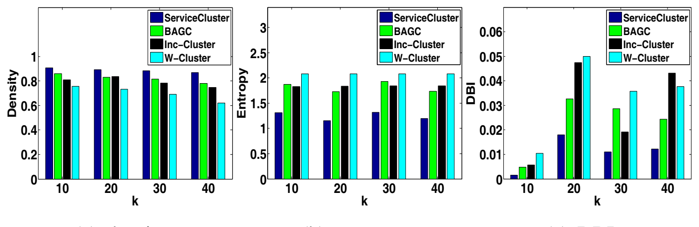
$$O(\{C_p\}_{p=1}^{k_i}, \nu) = \sum_{m=1}^{n} a_m e^{\nu^T p_m}, \nu = [\nu_1; \ldots; \nu_{(N+M)^2}], a_m \geq 0,$$
$$p_{mj} \geq 0, p_{mj} \in \mathbb{Z}, \sum_{j=1}^{(N+M)^2} p_{mj} \leq l, p_m = [p_{m1}; \ldots; p_{m((N+M)^2)}] \tag{12}$$
$$\text{s.t. } \sum_{j=1}^{(N+M)^2} e^{\nu_j} \leq (N+M)^2.$$

Notice that $O(\{C_p\}_{p=1}^{k_i}, \nu)$ is convex since it is a conic combination of convex functions $e^{\nu^T p_m}$. $\Theta = \{\nu | \sum_{j=1}^{(N+M)^2} e^{\nu_j} \leq (N+M)^2\}$ is convex since it has a less-than constraint and $\sum_{j=1}^{(N+M)^2} e^{\nu_j}$ is convex. We utilize the successive convex approximation method (SCA) [27] to maximize the convex objective on the convex set.

E. Splitting and Merging of Clusters

Due to the weight adjustment after each clustering iteration, we need to recalculate the unified random walk distance. This update operation essentially changes the shape and scale of dataset. We argue that a fixed k_i is no longer applicable to the dataset with changed shape. Thus we propose a dynamic adjustment method of k_i to identify a new suitable k_i to keep improving the clustering quality. Different from hierarchy-based clustering, our method may make k_i increase, decrease, or keep unchanged by splitting and merging current clusters after each iteration.

The cluster adjustment algorithm is presented in Algorithm 2. Instead of rediscovering local maxima of density based on the entire graph, we update local maxima with the prior knowledge of existing clustering result to continuously enhancing the clustering quality. It first calculates the density of each vertex on own cluster and find new potential local maxima in each cluster. The algorithm then splits each cluster into subclusters based on new DSD trees or paths. It figures out the density of local maxima and their ϵ-*neighborhood* on the entire graph. The density of vertices in the ϵ-*neighborhood* of a local maximum may be larger than the density of the local maximum due to the distance



(a) density (b) entropy (c) DBI

Figure 2. Cluster Quality on BSBM 10,000 Services

update. If the densest vertex in the ϵ-*neighborhood* has a larger density than the local maximum, then we merge two subclusters, where the densest vertex and the local maximum stay in, into a new cluster until all subclusters are scanned.

F. Clustering Algorithm

By assembling different parts, our heterogeneous network partitioning algorithm is presented in Algorithm 3. ServiceCluster consists of five main tasks: (1) initialization of k_i, (2) vertex assignment, (3) centroid update, (4) weight optimization, and (5) cluster adjustment and update of k_i, each with the goal of maximizing the clustering objective. Tasks (2)-(3) are common to partitioning clustering algorithms. The other three tasks are the novelty of this work.

VI. Experimental Evaluation

We have performed extensive experiments to evaluate the performance of ServiceCluster on real graph datasets.

A. Experimental Datasets

We modify the BSBM data generator [28] and create a dataset with $246,161$ triples where "Provides" is used to model the relationship between "Service" and "Provider"s providing them, while an instance of "Service" has multiple instances of properties "Capability", "Function" and "Type", and an instance of "Provider" contains multiple instances of properties "Feature" and "Type". There are totally $10,000$ "Service" instances and $3,628$ "Provider" instances with 10 "Type" instances and 5 instances of "Capability", "Function" and "Feature", respectively.

We extract the Artist-Work subset of the DBpedia data with $40,604$ artists with five kinds of identities [1], $136,048$ works from twelve kinds of areas [2], sixteen types of links [345], and four kinds of attributes [6]. We build a heterogeneous network where entity vertices represent artists or works, attribute vertices denote entity's attributes, entity edges represent the relationship between entities, attribute edges specify the interaction between entities and attributes.

We use a subset of the DBLP bibliography data with $200,000$ highly prolific authors and associated conferences. We build a heterogeneous network where vertices represent

[1]The artist type: Actor, Comedian, ComicsCreator, MusicalArtist, Writer.

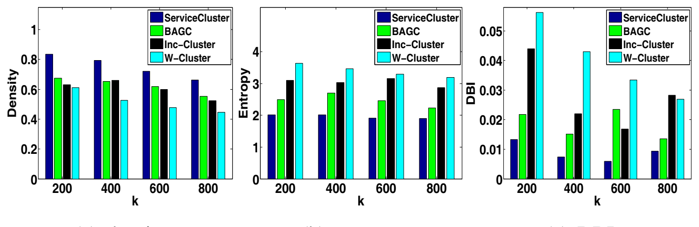
[2]The work type: Album, Book, ComicsCharacter, FictionalCharacter, Film, Magazine, Musical, Newspaper, Single, Song, TelevisionEpisode and TelevisionShow.

[3]The link type between artists: associatedMusicalArtist, influenced, influencedBy.

[4]The link type between works: album, basedOn.

[5]The link type between artists and works: artist, author, creator, director, editor, lyrics, musicalArtist, musicBy, producer, starring, writer.

[6]The attribute type: genre, literaryGenre, occupation, type.



(a) density (b) entropy (c) DBI

Figure 3. Cluster Quality on DBpedia 40,604 Artists

authors and conferences, links represent the number of coauthor works or the number of author's works on conference, and two relevant attributes: *prolific* and *primary topic*.

B. Comparison Methods and Evaluation

We compare **ServiceCluster** with two recently developed representative graph clustering algorithms, **BAGC** [19] and **Inc-Cluster** [17], and one baseline clustering algorithm, **W-Cluster**. ServiceCluster is our proposed algorithm which not only incorporates multiple types of entities, attributes and links into a unified distance model but also continuously enhances the clustering quality by simultaneously refining the link weights, the k value and the cluster centroids. Other three algorithms only integrate structural and attribute information to produce a clustering result. BAGC constructs a probabilistic inference model to capture both structural and attribute aspects. Inc-Cluster combines both structural and attribute similarities in the clustering decisions by estimating the importance of attributes. W-Cluster combines structural and attribute similarities with the equal weighting factors.

Evaluation Measures We use three measures to evaluate the quality of clusters $\{C_l\}_{l=1}^{k_i}$ generated by different methods. The definitions of the metrics are given as follows.

$$density(\{C_l\}_{l=1}^{k_i}) = \sum_{j=1}^{k_i} \frac{|\{(v_p, v_q)|v_p, v_q \in C_j, (v_p, v_q) \in E\}|}{|E|} \tag{13}$$

$$entropy(\{C_l\}_{l=1}^{k_i}) = \sum_{p=1}^{M} \frac{\omega_{ip}}{\sum_{q=1}^{M} \omega_{iq}} \sum_{j=1}^{k_i} \frac{|C_j|}{|V_i|} entropy(A_p, C_j) \tag{14}$$

where ω_{ip} is the weight between the objective entities V_i and the attribute A_p, $entropy(A_p, C_j) = -\sum_{m=1}^{m_p} p_{pjm} log_2 p_{pjm}$, m_p is the number of A_p's values and p_{pjm} is the percentage of entities in cluster C_j which have m^{th} value on A_p. $entropy(\{C_l\}_{l=1}^{k_i})$ measures the weighted entropy from all attributes over k_i clusters.

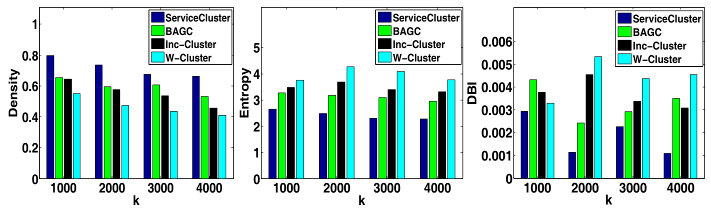
Davies-Bouldin Index (DBI) measures the uniqueness of clusters with respect to the unified similarity measure.

$$DBI(\{C_l\}_{l=1}^{k_i}) = \frac{1}{k_i} \sum_{p=1}^{k_i} max_{q \neq p} \frac{d(c_p, c_q)}{\sigma_p + \sigma_q} \tag{15}$$

where c_x is the centroid of C_x, $d(c_p, c_q)$ is the similarity between c_p and c_q, σ_x is the average similarity of entities in C_x to c_x.

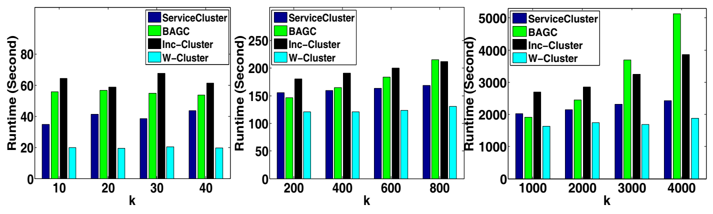
C. Cluster Quality Evaluation

Figures 2-4 show the quality comparison on three datasets with different k values. To make a fair comparison among different methods, we use a version of ServiceCluster with fixed k to compare other methods with the same k. Figure 2 (a) shows the density comparison on BSBM 10,000 Services by varying the number of clusters $k = 10, 20, 30, 40$. The



(a) density (b) entropy (c) DBI

Figure 4. Cluster Quality on DBLP 200,000 Authors



(a) BSBM 10,000 (b) DBpedia 40,604 (c) DBLP 200,000

Figure 5. Clustering Efficiency

density values by ServiceCluster, BAGC and Inc-Cluster remain 0.74 or higher even when k is increasing. This demonstrates that these methods can find densely connected components. However, ServiceCluster achieves a much higher density than other methods since it not only utilizes the link information between the objective entities but also integrates the interaction among the objective entities, other entities and relevant properties to improve the clustering quality. The density values by W-Cluster is relatively lower, in the range of 0.62-0.75 with increasing k, showing that the generated clusters have a very loose intra-cluster structure.

Figure 3 (b) shows the entropy comparison on BSBM 10,000 Services with $k = 10, 20, 30, 40$. ServiceCluster has the lowest entropy, while other three algorithms have a much higher entropy, since it not only considers the interaction between the objective entities and their associated attributes but also the interplay between the relevant entities and their attributes. Different from other iterative local maximization approaches, ServiceCluster generates the near global optimal weight assignment for each kind of links by directly solving the maximization problem of clustering objective.

Figure 3 (c) shows the DBI comparison on BSBM 10,000 Services with different k values. ServiceCluster has the lowest DBI of 0.003-0.018, while other methods have a much higher DBI than ServiceCluster. This demonstrates that ServiceCluster can obtain both high intra-cluster similarity and low inter-cluster similarity. This is because ServiceCluster incorporates multiple types of entities, attributes, and links with the near global optimal weight assignment. It fully utilizes the connection among the objective entities and other relevant entities, and the interaction between the entities and their attributes such that the generated clusters have not only similar collaborative patterns but also similar interplay patterns with relevant entities and associated attributes.

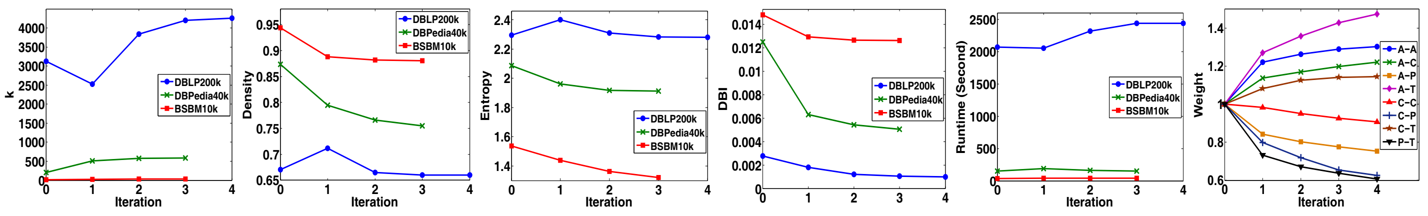
Similar trends are observed for the quality comparison on other two datasets in Figures 3 and 4: ServiceCluster achieves the highest density values (>0.66) but the lowest entropy around 1.89-2.64, which is obviously better than the other methods (>2.24). As k increases, the entropy by ServiceCluster remains stable, while the density of ServiceCluster decreases. In addition, ServiceCluster achieves the lowest DBI (0.001-0.013) among different methods.

D. Clustering Efficiency Evaluation

Figures 5 (a) and (b) show the clustering time on three real datasets respectively. ServiceCluster outperforms all other algorithms in all experiments. We make the following observations on the runtime costs of different methods. First, W-Cluster with a fixed weight assignment is obviously better than other methods since it computes the random walk distance and does graph clustering only once. After each clustering iteration, ServiceCluster and Inc-Cluster need to incrementally update the random walk distance matrix. The cost of incremental distance update is relatively trivial in comparison with recalculating the matrix from scratch. Second, ServiceCluster is much faster than BAGC and Inc-Cluster since it figures out the near global optimal weight assignment by using the successive convex approximation method. Other two local maximization methods by using iterative probabilistic influence or majority vote strategy often converge to a local maximum, even converge to a local minimum or cycle between two points such that they need more iterations to terminate the clustering process. Third, BAGC is much slower than and Inc-Cluster when facing large k values since it is hypersensitive to k. Although BAGC does not need to repeatedly compute the distance matrix, it needs to iteratively update a clustering membership matrix with the size of $n \times k$ and lots of temporary matrices or interim variables such as $\widetilde{\xi}$, $\widetilde{\gamma}$, $\widetilde{\mu}$, $\widetilde{\nu}$ and $\widetilde{\beta}$. As a result, its computational cost is proportional to n^2k^2 such that it may not work well when facing large k values.

E. Clustering Convergence

Figures 6 (a)-(f) show the tendency of clustering convergence of ServiceCluster along with the dynamic update k at each iteration on three datasets respectively. Figure 6 (a) shows the convergence trend of k on different datasets. The k value converges very quickly, usually in four to five iterations. This demonstrates the efficiency of the algorithm. Figures 6 (b) and (c) show how the clustering quality progresses along with the dynamic update k. We know that both density and entropy may decrease with increasing k. Their decreasing trends are highly correlated to some measure of diffuseness such as average cluster radius (or diameter). There may exist a critical point in the plane with k as x-axis and average cluster radius as y-axis, i.e., the average cluster radius decreases quickly as soon as k falls below the critical point but drops slowly as long as k remains at or above the critical point. When the k value arrives at or above the critical point in enough clustering iterations, both density and entropy finally converge to stable values. Figure 6 (d) shows the convergence tendency of DBI along with the k values. A low DBI value identifies a clustering with high intra-cluster similarity and low inter-cluster similarity.



(a) k (b) density (c) entropy (d) DBI (e) Runtime (f) Weight

Figure 6. Clustering Convergence on Different Datasets

ServiceCluster exhibits superior performance on all datasets in terms of the DBI measure. The convergence tendency of DBI is consistent with the convergence trend of k on each dataset. Figures 6 (e) shows the curve of running time keeps relatively stable when the k value arrives at or above the critical point. Figure 6 (f) shows the trend of weight updates on DBLP 200,000 Authors along with the dynamic update k where letters "A" and "C" represent two types of entities: *author* and *conference* respectively, and letters "P" and "T" denote two kinds of associated attributes: *prolific* and *primary topic* respectively.

VII. Conclusion

We have presented ServiceCluster, a novel heterogeneous service network clustering framework. First, we integrate multiple types of entities, attributes and links with different semantics into a unified random walk distance model. Second, we design a DSD method to naturally produce initial k and initial centroids simultaneously. Third, a dynamic learning approach is proposed to refine the link weights, the k value and the cluster centroids to constantly improve the clustering quality.

Acknowledgment

This material is based upon work partially supported by the National Science Foundation under Grants IIS-0905493, CNS-1115375, IIP-1230740, and a grant from Intel ISTC on Cloud Computing.

References

[1] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann. Dbpedia - a crystallization point for the web of data. In *Web Semantics*, 7(3), 154–165, 2009.

[2] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering wsdl documents to bootstrap the discovery of web services," in *ICWS*, 2010, pp. 147–154.

[3] M. Aznag, M. Quafafou, N. Durand, and Z. Jarir, "Multiple representations of web services: Discovery, clustering and recommendation," in *ICWS*, 2011, pp. 748–749.

[4] S. Dasgupta, S. Bhat, and Y. Lee, "Taxonomic clustering and query matching for efficient service discovery," *ICWS*'11.

[5] H. Q. Yu, X. Zhao, S. Reiff-Marganiec, and J. Domingue, "Linked context: A linked data approach to personalised service provisioning," in *ICWS*, 2012, pp. 376–383.

[6] Y. Zhou, L. Liu, C.-S. Perng, A. Sailer, I. Silva-Lepe, and Z. Su, "Ranking services by service network structure and service attributes," in *ICWS*, 2013.

[7] M. Aznag, M. Quafafou, and Z. Jarir, "Leveraging Formal Concept Analysis with Topic Correlation for Service Clustering and Discovery," in *ICWS*, 2014, pp. 153–160.

[8] D. Skoutas, D. Sacharidis, A. Simitsis, and T. Sellis, "Ranking and clustering web services using multi-criteria dominance relationships," *TSC*, 3(3), pp. 163–177, 2010.

[9] H. Xiao, Y. Zou, J. Ng, and L. Nigul, "An approach for context-aware service discovery and recommendation," in *ICWS*, 2010, pp. 163–170.

[10] M. Almulla, K. Almatori, and H. Yahyaoui, "A qos-based fuzzy model for ranking real world web services," *ICWS*'11.

[11] G. Liu, Y. Wang, M. A. Orgun, and H. Liu, "Discovering trust networks for the selection of trustworthy service providers in complex contextual social networks," in *ICWS*, 2012.

[12] B. Kumara, I. Paik, and W. Chen, "Web-service Clustering with a Hybrid of Ontology Learning and Information-retrieval-based Term Similarity," in *ICWS*, 2013, pp. 340–347.

[13] M. Shiga, I. Takigawa, H. Mamitsuka. A spectral clustering approach to optimally combining numericalvectors with a modular network. In *KDD*, 2007.

[14] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: a structural clustering algorithm for networks. In *KDD*, 2007.

[15] V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: Applications to community discovery. In *KDD*, 2009.

[16] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. In *VLDB*, 718–729, 2009.

[17] Y. Zhou, H. Cheng, and J. X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*, 2010.

[18] M. Ji, J. Han, and M. Danilevsky. Ranking-based classification of heterogeneous information networks. In *KDD*, 2011.

[19] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. *SIGMOD*'12.

[20] Y. Sun, C. C. Aggarwal, and J. Han. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *PVLDB*, 5(5):394–405, 2012.

[21] W. Cheng, X. Zhang, Z. Guo, Y. Wu, P. Sullivan, and W. Wang. Flexible and robust co-regularized multi-domain graph clustering. In *KDD*, 2013.

[22] Y. Zhou and L. Liu. Social influence based clustering of heterogeneous information networks. In *KDD*, 2013.

[23] Y. Zhou and L. Liu. Activity-edge centric multi-label classification for mining heterogeneous information networks. In *KDD*, 2014.

[24] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *SODA*, pages 1027–1035, 2007.

[25] G. Hamerly and C. Elkan. Learning the k in k-means. In *NIPS*, 2003.

[26] L. Kaufman and P. J. Rousseeuw. Clustering by means of medoids. In *Statistical Data Analysis based on the L1 Norm*, 405–416, 1987.

[27] F. Hillier and G. Lieberman. Introduction to Operations Research. In *McGraw-Hill College*, 1995.

[28] C. Bizer and A. Schultz, "The berlin sparql benchmark," *IJSWIS*, 5(2), pp. 1–24, 2009.

[29] T. Hofmann. Probabilistic latent semantic indexing. *UAI*'99.