SCOOP: Automated Social Recommendation in Enterprise Process Management

Huiming Qu Jimeng Sun Hani T. Jamjoom
IBM TJ Watson Research Center
{hqu, jimeng, jamjoom}@us.ibm.com

Abstract

The interplay between labor arbitrage and consistent service delivery in IT outsourcing continues to drive business process standardization. New emerging standards, like ITIL, is defining an industry-wide taxonomy for IT service management. These standards are often at a high level and require substantial investment from service providers to define and implement these standards across the various low-level processes they offer.

This paper presents a process management system, called Cyano, that uses social networks and recommendation to greatly increase the effectiveness of process capture and knowledge maintenance. We particularly focus on Cyano's social recommendation engine, called SCOOP*, which utilizes the intrinsic graph property of process content for recommendation. More specifically, SCOOP maintains a user-process interaction graph G and computes the user-to-user similarity scores using the random walk with restart on G. Finally, we evaluate SCOOP in the context of a large-scale deployment of Cyano, with thousands of processes and users.*

1 Introduction

Nowadays, IT service providers deliver a plethora of solutions covering a broad spectrum of services, e.g., help-desk, IT infrastructure maintenance, technical support, data storage, etc. From service providers' point of view, it is of paramount importance to provide these services through standardized processes. These processes, however, are typically defined at different information granularities, with lower-level processes often referred to as *best practices*. Clearly, the goal is to have every Subject Matter Expert (SME) follow the best practices while providing/implementing a specific service.

Following the best practices is not always feasible or cost efficient. In order to do so, all the processes have to be documented and maintained into practice guidelines by a few experts. They are the *content creators*. All SMEs then have to follow these guidelines. They are the *content consumers*. It is very costly to keep the guidelines up-to-date and constantly push them into practice. This problem is extremely prominent in the diverse and dynamic environment of IT service delivery. Here, thousands of processes have to be constantly maintained and enforced on a global basis.

In this paper, we present an alternative solution for business process management. Instead of having a few experts being the process creators, a Web-based application is provided for all SMEs to collaborate in standardizing and optimizing processes. In particular, we developed and deployed a process collaboration wiki, called *Cyano*, that is being used to publish hundreds of processes and used by thousands of SMEs. This wiki enables its users to become content creators in addition to content consumers. Basically, SMEs can follow the state of the art to do their daily job and, more importantly, contribute their knowledge and experience through creating, updating, and annotating the relevant business processes. Compared to traditional approaches, social collaboration not only reduces the training and maintenance cost, it also increases productivity of SMEs by simplifying the process of knowledge creation. Due to the content's immediate relevance, most of SMEs are willing to serve as both process consumers as well as creators. This role shift is revolutionary because it is now possible to leverage collective knowledge from all SMEs on a day-to-day basis.

As will other collaborative knowledge authoring platforms, the capture mechanism is only the first step to realize the true potential of social networks in this space. In fact, we argue that the true knowledge about best practices can be better conveyed through social interaction among SMEs. Mirroring similar goals to successful social networks, the key is to enable better communication channels between users. To address this problem, we present *SCOOP*, a social rec-

978-0-7695-3283-7/08 $25.00 © 2008 IEEE
DOI 10.1109/SCC.2008.153

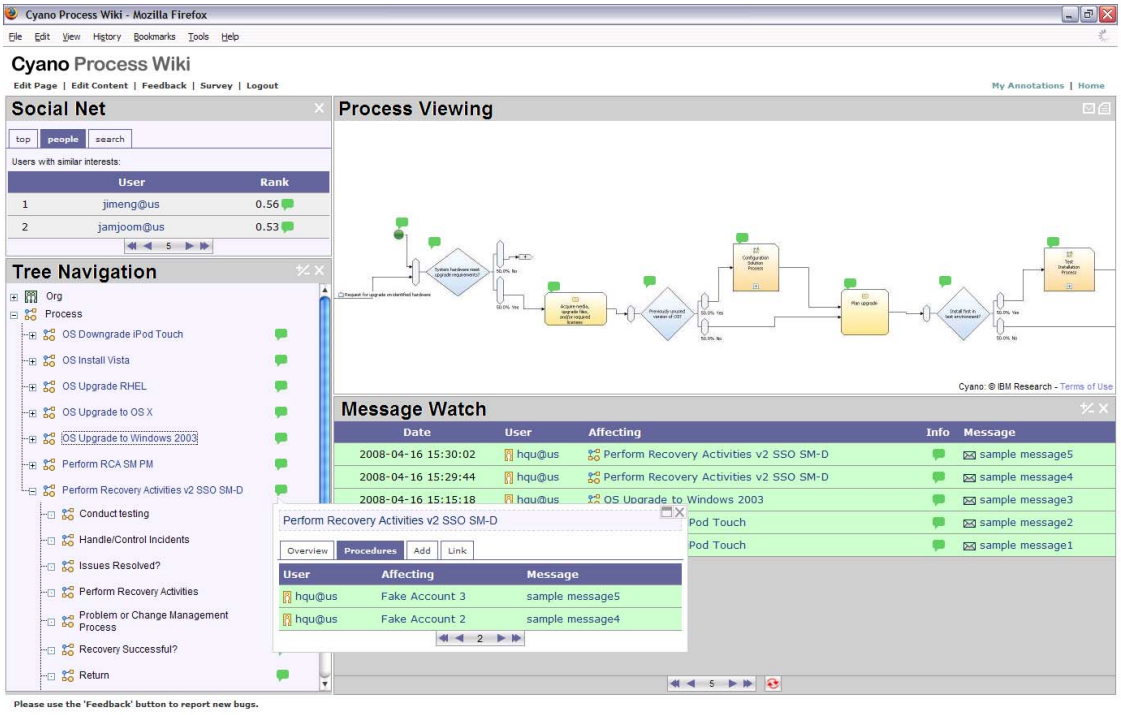


Figure 1. Process Management Web Screen-shot

ommendation system that provides customized recommendations based on the process knowledge generated by individual SMEs. SCOOP utilizes the network connectivity between users (the content creators) and tasks (the content) to compute the personalized neighborhood for each SME. The neighborhoods are constructed using "random walk with restart" on the user-task network, which is an elegant and powerful method to measure a network similarity. We have deployed and evaluated SCOOP on a real enterprise process wiki with about 10,000 users and 6,000 processes. We also evaluate the scalability and robustness of SCOOP and illustrate its superiority over two popular similarity measures.

The main contributions of this paper are as follows:

- We introduce a business process management system based on social networks;
- We propose a novel social recommendation system, SCOOP, to personalize user neighborhoods for collaboration;
- We deploy SCOOP to a production environment and confirm its success based on real data and usages.

The remainder of the paper is organized as follows. Section 2 introduces our Cyano for business process management. Section 3 presents SCOOP's algorithms for social recommendation. In Section 4, we evaluate the effectiveness and efficiency of SCOOP in a production environment with real data. We discuss the related work in Section 5, and conclude in Section 6.

2 Enterprise Social Network for Process Management

In this section, we first present process modeling capabilities in Cyano. Second, we provide an overview of the social networking functionalities for enabling collaborations.

2.1 Process Modeling and Management

Cyano is implemented using LAMP (Linux-Apache-MySQL-PHP) with several AJAX libraries to support an interactive experience. Figure 1 shows a screen-shot of Cyano. Here, we give a high-level overview its core functionalities.

Process as a flowgraph. A process model is a sequence of tasks to achieve certain business objectives. These models are treated as base templates that can be annotated. We represent a process as a flowgraph

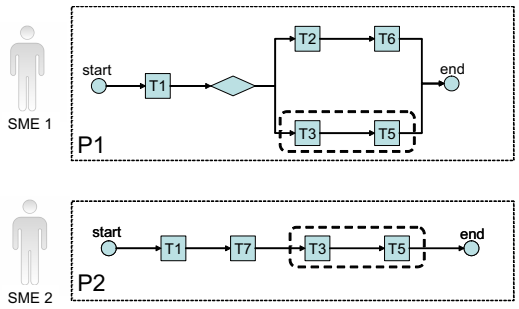


Figure 2. Process Example. Two processes P1 and P2 share common tasks T3 and T5. Often different SMEs operate on them without knowing the commonality.

of tasks, e.g., Figure 2; a task is an atomic unit of actions, e.g., installing Linux OS is a task in the process for building a IBM z-Series server. Our system accepts process models authored through a variety of tools, including Visio, Dia, and WebSphere Business Modeler. A process shown in the Process Viewing window can be dragged around, zoomed in and out. Each task has a "bubble" on top of it that can be expanded to display detailed information and to add annotations.

Annotating processes/tasks. SMEs can annotate each process or task. An annotation typically captures additional attributes relating to a process or a task. Annotation content vary from prescriptive instructions to account-specific variations. Annotations also act as bases for discussion threads, which are internally treated as annotating an annotations. Each annotation is viewed as a link between the user and the task/process. These annotations are the base for our social recommendation system as we will describe shortly.

Social incentives and reputation mechanism. As with other community-based authoring systems, Cyano leverages a simple social incentive and reputation mechanism to maintain the quality of authored content. Basically, users co-rate other users' content. Authorship and ratings are used to compute top authors, which is visible to the entire community.

In summary, process information is dynamically managed through a web portal with an underlying annotation database. This system provides a dynamic platform for both process creation and re-engineering.

2.2 Social Recommendation in User-Process networks

In addition to the common functionalities such as browsing, commenting, rating of tasks and users, a set of social recommendation capabilities supported by the SCOOP engine are also implemented in the system.

Knowledge sharing on common tasks. Usually, SMEs are organized according to their specialties and focus on specific processes in their domains. For example, in Figure 2, SME 1 and 2 work on process P1 and P2, respectively. Multiple processes can share the same tasks. In Figure 2, task T3 and T5 are common in both process P1 and P2. However, this commonality is often unknown to most of SMEs, since they tend to focus only on the processes they work on. As we show shortly, SCOOP specifically makes use of this latent connection of common tasks for social recommendation, which infers the user-user relationship through user-process networks.

Search similar users. One core functionality in SCOOP is to find relevant users given a specific user, and rank the users according to their relevance. We call this step *neighborhood formation*. A typical use-case is that an SM want to consult other SMEs who work on similar tasks. Besides user neighborhood formation, SCOOP can also discover user-task and task-task relationship.

The challenge of neighborhood formation is to design an accurate and intuitive similarity measure between users based on their historical annotations. Here are some of design considerations:

a. SME a is closer to SME b than SME c if a and b share more common tasks.

b. SME a is closer to task T than SME b if a has more annotations on T.

c. SME a is closer on task T than SME b if b works on many tasks, while a only works on task T.

d. Task $T1$ is closers to SME a than task $T2$ if $T1$ has fewer user work on.

As we will show in Section 3, SCOOP can successfully handle all of the above scenarios with a simple and elegant formulation.

SCOOP engine. We implemented the SCOOP engine using Java 5. The SCOOP engine reads from the system's annotation database, performs neighborhood formation, and then store similar scores of users back

Symbol	*Description*
V_1	the set of k row nodes
V_2	the set of n column nodes
M	the k-by-n bipartite matrix
M^T	the transpose of M
M_A	$(k+n)$-by-$(k+n)$ adjacent matrix
P_A	$(k+n)$-by-$(k+n)$ Markov transition matrix
$\vec{r_a}$	the k-by-1 relevance score vector for $a \in V_1$
c	the restarting probability

Table 1. Symbol Table

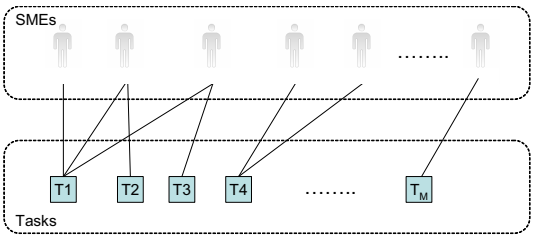


Figure 3. User-task Network. Two groups of nodes - users (SMEs) and tasks; each edge indicates the annotations of a SME on a task.

to the MySQL database. Essentially, the SCOOP engine is a stand-alone social recommendation layer to be deployed in other social network applications.

3 SCOOP: Social Recommendation

In this section, we first introduce the data model for the user-task network. Then, we present neighborhood search and connected subgraph computation on such network.

3.1 Data model

We now define our data model and terminology. A summary of the notations is introduced in Table 1.

The user-task network is viewed as a weighted and undirected bipartite graph $G = \langle V_1 \cup V_2, E, W \rangle$, where $V_1 = \{u_i | 1 \leq i \leq k\}$ are the user nodes and $V_2 = \{t_i | 1 \leq i \leq n\}$ are the task nodes, and $E = \{< v_1, v_2, w > | v_1 \in V_1, v_2 \in V_2, w \geq 0\}$ are the edges between users and tasks. The graph G is stored in a k-by-n matrix M,[1] where $M(i,j)$ is the weight w of the edge $< i, j >$. In SCOOP, we use the number of annotations from user i to task j as the weights, but the algorithm works for arbitrary weighting functions. For example, the graph in Figure 3 becomes the following matrix:

$$M_{k \times n} = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & 0 & \dots & 1 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 1 \\ \dots & & & & & \\ 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

The nodes in $V_1(V_2)$ are called row(column) nodes. Note that a column node links to a row node if the corresponding matrix element is nonzero. Moreover, row node a connects to another row node b if there is a column node c linking to both a and b. We call that path a *connection* between a and b through c.

We can easily construct the adjacency matrix M_A of G using M:

$$M_A = \begin{pmatrix} 0 & M \\ M^T & 0 \end{pmatrix} \quad (1)$$

In particular, $M_A(a,t)$ denotes the element at a-th row and t-th column in M_A.

Suppose we want to traverse the graph starting from the row node (user) u. The probability of taking a particular edge $< u, t >$ is proportional to the edge weight over all the outgoing edges from u. Intuitively, in the user-task network, it means that the importance of a task to a user is proportional to the number of annotations the user has posted on that particular task. More formally, the importance is measured by the Markov transition probability $P_A(u,t) = M_A(u,t) / \sum_{i=1}^{k+n} M_A(u,i)$. Therefore, the Markov transition matrix P_A of G is constructed as: $P_A = col\_norm(M_A)$, where $col\_norm(M_A)$ normalizes M_A such that every column sum up to 1.

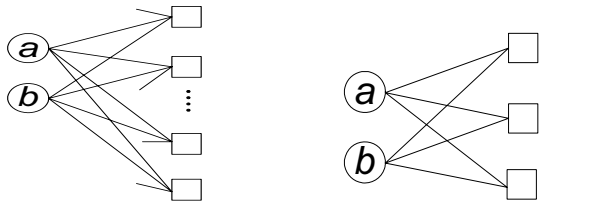
The main reasons to have M instead of working directly on M_A and P_A are the computational and storage savings. Next, we define the social recommendation algorithm in SCOOP.

3.2 Neighborhood search

Now we present the algorithm to compute the neighborhood for given a user. In particular, we want to answer the following question: Given a node $u \in V_1$, which nodes in V_1 are most related to u?

Given a row node $a \in V_1$, we want to compute a relevance score for each row node $b \in V_1$. The final result is a score vector consisting of the relevance scores of all k users to a.

[1] In practice, we adopt the sparse matrix representation where the storage space is proportional to the number of non-zero elements in the matrix.



(1) Many connections between a and b (2) Exclusive connections between a and b

Figure 4. Neighborhood Scenarios

Intuition. Intuitively, we do multiple random walks starting from a, and count the number of times that we visit each $b \in V_1$. These counts reflect the relevance of those nodes to a. The probability of visiting $b \in V_1$ from a is the relevance score we want to obtain. In the following, we list some scenarios on which the row nodes have high relevance scores.

As shown in Figure 4(2), b usually has a high relevance score to a if (1) b has many connections to a as shown in Figure 4(1); or (2) the connections only involve a and b. Scenario (1) is obvious because the row nodes b and a have many connections through the columns nodes, which indicates the strong relevance between b and a. Scenario (2) is less obvious. The intuition is that the connection that only links a and b brings more relevance between a and b than the connections linking a, b and other nodes. The relevance score is not only related to the number of connections, but also to the number of nodes involved in the connections. One observation is that the node with the highest relevance score is not necessarily the one with most connections to a. The reason is that those connections link to nodes other than a and b as well. Thus, the relevance spreads out among many different nodes. Nevertheless, all the scenarios in Section 2.2 are well captured by our algorithm in spite of its simplicity.

Algorithms. First, we transform the input row node a into a $(k+n) \times 1$ query vector $\vec{q_a}$ with 1 in the a-th row and 0 otherwise. Second, we need to compute the $(k+n) \times 1$ steady-state probability vector $\vec{u_a}$ over all the nodes in G. Last, we extract the probabilities of the row nodes as the score vectors. Note that $\vec{u_a}$ can be computed by an iterated method from the following lemma.

Lemma 3.1. *Let c be the probability of restarting random-walk from the row node a. Then the steady-state probability vector $\vec{u_a}$ satisfies*

$$\vec{u_a} = (1-c)P_A\vec{u_a} + c\vec{q_a} \quad (2)$$

where P_A is already the column normalized.

Algorithm 1: NEIGHBORHOOD SEARCH

Input: node a, bipartite matrix M, restarting probability c, tolerant threshold E

1 Initialize $\vec{q_a} = 0$ except the a-th element is 1 ($q_a(a) = 1$)
2 construct M_A (see Equation 1) and $P_A = col\_norm(M_A)$
3 **while** $|\Delta\vec{u_a}|^3 > E$ **do**
4 $\quad \vec{u_a} = (1-c)P_A\vec{u_a} + c\vec{q_a}$
5 return $\vec{u_a}(1:k)$

The algorithm applies Equation 2 repeatedly until it converges. The actual computation of the algorithm can utilize the bipartite structure to have more saving [10]. More specifically, we do not materialize M_A and P_A for computional and storage efficiency. Instead, we modify Equation 2 as follows:

$$\vec{u_a} = (1-c)\begin{pmatrix} \texttt{col\_norm}(M)\vec{u_a}(k+1:k+n) \\ \texttt{col\_norm}(M^T)\vec{u_a}(1:k) \end{pmatrix} + c\vec{q_a} \quad (3)$$

where $\vec{u_a}(1:k)$ and $\vec{u_a}(k+1:k+n)$ are the vectors of first k and last n elements of $\vec{u_a}$, respectively. The relevance score $rs(a)$ is $\vec{u_a}(1:k)$. If we compute the relevance scores for all the nodes, we have a similarity matrix S.

The saving is significant when the number of rows k and the number of columns n differ a lot. Therefore, Equation 3 is always recommended in practice, while Equation 2 is only for demonstrating the concept.

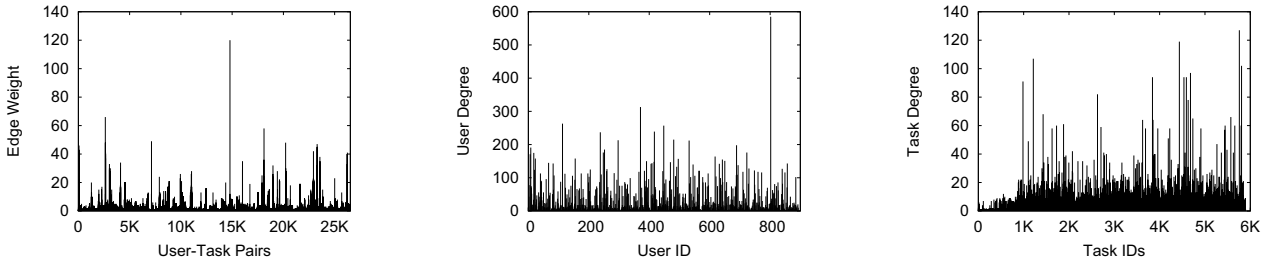
4 Experimental Evaluation

We evaluate the SCOOP system from three different aspects. First, we introduce the data characteristics of the user-task network, which provides intuition and insights of the proposed algorithm. Second, we compare SCOOP similarity score function with two standard similarity measures, (1) Jaccard coefficient and (2) Cosine similarity. Third, we illustrate the impacts of different algorithm- and data-specific parameters.

4.1 Data characterization

Data. SCOOP has been deployed in Cyano Process Wiki. It currently consists of 5,896 tasks and 10,000

[2] c is set to 0.15 for all experiments.



(a) Edge weight (b) User degree (c) Task degree

Figure 5. Data Distribution. The degree and edge weight distributions are all very skewed.

active users (among which 893 users annotated the tasks). From the wiki's database, we construct a 893-by-5,896 bipartite network G. The total number of annotations is 49,336 and the number of distinct user-task pairs (i.e., the number of edges) $|E| = 26,531$.

Data distribution. Figure 5 shows the data properties of the user-task network. In particular, Figure 5(a) plots the edge weight (the number of annotations per user-task pair). The mode of edge weights is 1, meaning that most of users only posted one annotation on any particular task. However, occasionally some user-task pairs have many annotations piled up when the tasks are complex and/or when users are commenting on other users annotations on the same tasks.

Another important property is node degree, which is the number of edges incident to a particular node. Here we have two types of nodes, users and tasks. The user degree distribution is plotted in Figure 5(b). Depending on the roles, the number of annotations from different types of users can vary considerably. For example, a DBA can bulk load many annotations at once. Similar pattern is observed in the task degree distribution as shown Figure 5(c). Some complex task such as "Monitoring infrastructure" attracted over 100 users to annotate.

4.2 Comparison to Existing Methods

The main advantage of SCOOP compared to standard similarity measures is its ability to (1) identify hidden connections and (2) provide ranking based on the network connectivity properties. To elaborate on these two points, let us first introduce two popular similarity measures:

- **Jaccard coefficient** measures the similarity of two vectors $\mathbf{x}$ and $\mathbf{y}$ by the number of the overlapping elements, namely, $\sum_i x_i y_i$.

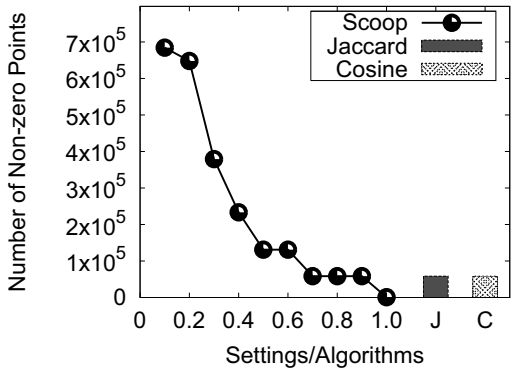


Figure 6. Number of non-zero points vs. Algorithms (SCOOP with various settings of restart probability, c). SCOOP identifies more hidden connections than Jaccard and Cosine except the extreme case.

- **Cosine similarity** is a normalized version of Jaccard coefficient by the norm of $\mathbf{x}$ and $\mathbf{y}$, namely, $\frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2 y_2^2}}$.

Both methods can be applied to the row vectors of the user-task matrix, M_A, to generate user-user similarity scores. However, they can only measure the similarity of two users if they are directly connected through a task. In our case, the user-task matrix is very sparse, as a result, most of user-user pair have 0 score due to the lack of direct connections.

Figure 6 shows the number of non-zero scores of three different methods: (1) SCOOP with the restart probability varying from .1 to 1, (2) Jaccard coefficient, and (3) Cosine similarity. SCOOP has substantially more nonzero scores because of its ability to utilize in-

direct links among users. One interesting observation is that the number of nonzero scores drops as the restart probability increases. This is because a large restarting probability, c, restricts the scope of the search to the starting point. When c equals to 1, meaning that the random walk always jumps back to the starting point, SCOOP finds the smallest neighborhood.

4.3 Parameter Selection and Impact

The SCOOP algorithm involves three different parameters:

- **Restart probability** c is the probability going back to the query node in the random walk process. Our default is 0.3.
- **Convergence threshold** E determines if the score vector is converged based on the difference between consecutive iterations. Our default is 0.1.
- **Network size** M the number of annotations in the network.

To quantify the performance, we use the following metrics:

- **Time** T refers to the elapsed time in computing the similarity scores.
- **top-k precision** is the overlapping percentage between the top-k highest ranked neighbors on a given user. It measures the difference on a single user across two experiment settings.
- **Average top-k precision** for two experiments is the average top-k precision for all users.

c	.1	.2	**.3**	.4	.5
prec	.80	.88	**1**	.94	.90
time(min)	10	7.16	**4.4**	3.03	2.32

Table 2. Varying c. Large c leads to short computational time; compared to $c = .3$, the average top-10 precision is fairly stable across different values of c.

Effects of restart probability c. The restart probability c affects the convergence speed. As shown in the third row of Table 2, large c values speed up the algorithm's convergence because the increase of c reduces the search scope. Typically, we only care about the most similar/relevant users to a query user. Therefore, we compute the average top-10 precision of $c = .1$ to .5, comparing against $c = .3$. As shown in the second row of Table 2, the precision is high across all c, which means the neighborhood of a user is fairly robust to the change of c.

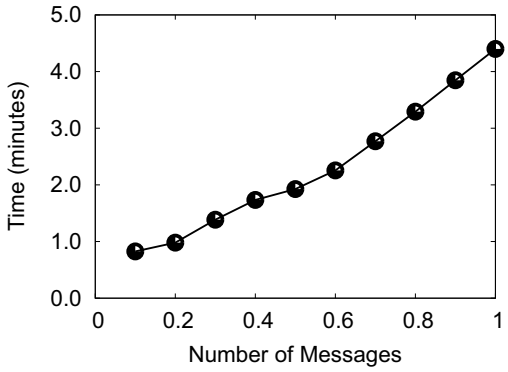


Figure 7. Time vs. Network size (percentage of annotations included). SCOOP is scalable to the network growth.

Effect of convergence threshold E. The convergence threshold E is another parameter in SCOOP. As shown in the second row of Table 3, the average top-10 precision varies little across a wide range of E; on the other hand, computational time drops dramatically as we increase the threshold E. These two observations suggest that we can safely choose a relatively large E with small computational cost to achieve nearly the same result as a small E.

E	.0001	.001	.05	.1	.2	.3
prec	1	.953	.954	.940	.928	.938
time(min)	11.75	8.92	6.58	4.4	4.22	3.85

Table 3. Varying E. We compare top-10 precision of various E compared to the base $E = .0001$. The precision is consistently high across all c. The computational time however reduces significantly as c increases.

Effect of network size. Figure 7 shows the computational time (y-axis) vs. the number of annotations (as percentages to the total annotations, x-axis). The scale is linear, confirming that SCOOP is scalable to network growth.

5 Related Work

5.1 Similarity and Ranking for Networks

The SCOOP algorithm is based on random-walk with restart, which relates to Page-Rank [1] and its variants. Page-Rank [1] learns the ranks of web pages using the iterated power method on web graph M (adjacency matrix of the entire graph). The ranks of all webpages are cast as an N-dimensional vector, and then the fixed point is found for the following equation: $\vec{r} = (1-\alpha)M \times \vec{r} + \alpha\vec{p}$, where the α is the damping factor and $\vec{p} = [\frac{1}{N}]N \times 1$. Thus, there is an uniform prior on all the web pages. In order to deal with personalized query, Topic-Sensitive PageRank [4] increases the importance of certain web pages by putting non-uniform weights for $\vec{p}$. Similar random-walk approaches have been used into many other domains: Mixed Media Graph(MMG) on bio-image retrieval [8], SimRank on web graphs [5], neighborhood formation on bipartite graphs [9]. To our knowledge, SCOOP is the first attempt to apply random-walk with restart to IT service application.

5.2 Business Process Management

Business process management is a broad area in enterprise operation and management. A number of commercial systems have been used heavily for various purpose, such as IBM MQSeries for workflow and message management, SAP or Oracle for process and data management. Many research efforts have been focusing on framework design and process integration, transformation, and verification. Examples of such research include process transformation and integration [3, 2], and verification framework for web service [7, 6].

One work on this area is by Yang et al. [11] in which they propose a social network framework for Web 2.0 application. Our work has a different focus, namely, on leveraging social network and data mining for IT outsourcing application.

6 Conclusion

As it becomes an inevitable trend for global business operations, IT outsourcing also poses great challenges on service providers. To achieve efficient and effective service delivery, we developed and deployed Cyano, a business process management system for IT delivery through social networking. The system enables active participation from all SMEs to create and to maintain a dynamic process knowledge base through intuitive graphical wiki. We also developed SCOOP to automatically leverage the latent connections of different processes for social recommendation. Extensive evaluation of SCOOP is provided based on real data in the system.

Future work includes the development of online SCOOP algorithm to efficiently handle updates and the automatic interpretation of recommendation results.

References

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[2] C.-M. Chituc, C. Toscano, and A. L. Azevedo. Collaborative business processes integration and management - lessons learned from industry. In *IEEE SCC*, pages 451–457, 2007.

[3] K. Fujiwara, B. Ramachandran, A. Koide, and J. Benayon. Business process transformation wizard: a bridge between business analysts and business process transformation technology. In *IEEE SCC*, pages 83–90, 2007.

[4] T. Haveliwala. Topic-sensitive pagerank. In *WWW*, 2002.

[5] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD*, 2002.

[6] N. Luo, J. Yan, and M. Liu. Towards efficient verification for process composition of semantic web services. In *IEEE SCC*, pages 220–227, 2007.

[7] S. Moser, A. Martens, K. Gorlach, W. Amme, and A. Godlinski. Advanced verification of distributed ws-bpel business processes incorporating cssa-based data flow analysis. In *IEEE SCC*, pages 98–105, 2007.

[8] J.-Y. Pan, H.-J. Yang, P. Duygulu, and C. Faloutsos. Automatic multimedia cross-modal correlation discovery. In *KDD*, 2004.

[9] J. Sun, S. Papadimitriou, and C. Faloutsos. Online latent variable detection in sensor networks. In *ICDE*, pages 1126–1127, 2005.

[10] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM*, pages 418–425, 2005.

[11] S. J. H. Yang, J. Zhang, and I. Y. L. Chen. Web 2.0 services for identifying communities of practice through social networks. In *IEEE SCC*, pages 130–137, 2007.