

# Analyses for Service Interaction Networks with applications to Service Delivery \*

S. Kameshwaran <sup>†</sup>      Sameep Mehta <sup>‡</sup>      Vinayaka Pandit <sup>‡</sup>      Gyana Parija <sup>‡</sup>  
Sudhanshu Singh <sup>§¶</sup>      N. Viswanadham <sup>†</sup>

## Abstract

One of the distinguishing features of the services industry is the high emphasis on people interacting with other people and serving customers rather than transforming physical goods like in the traditional manufacturing processes. It is evident that analysis of such interactions is an essential aspect of designing effective and efficient services delivery. In this work we focus on learning individual and team behavior of different people or agents of a service organization by studying the patterns and outcomes of historical interactions. For each past interaction, we assume that only the list of participants and an outcome indicating the overall effectiveness of the interaction are known. Note that this offers limited information on the mutual (pairwise) compatibility of different participants. We develop the notion of *service interaction networks* which is an abstraction of the historical data and allows one to cast practical problems in a formal setting. We identify the unique characteristics of analyzing service interaction networks when compared to traditional analyses considered in social network analysis and establish a need for new modeling and algorithmic techniques for such networks. On the algorithmic front, we develop new algorithms to infer attributes of agents individually and in team settings. Our first algorithm is based on a novel modification to the eigen-vector based centrality for ranking the agents and the second algorithm is an iterative update technique that can be applied for subsets of agents as well. One of the challenges of conducting research in this setting is the sensitive and proprietary nature of the data. Therefore, there is a need for a realistic simulator for studying service interaction networks. We present the initial version of our simulator that is geared to capture several characteristics of service interaction networks that arise in real-life.

## 1 Introduction

One of the distinguishing features of the services industry is the high emphasis on people interacting with other people and serving customers rather than transforming physical goods in the process. The focus of this paper is on analyzing patterns of interactions in services delivery to help understand the impact of individuals and their interactions on the quality/effectiveness. A typical service creation can be thought of as a *workflow* consisting of various stages of specialized activities. Therefore, service creation involves people, each performing the specialized tasks of the stage that he or she is responsible for, and communicating with others who are performing related/dependent activities. It should be emphasized that we use the term workflow in a broader sense than the specific notions they carry in specialized literature such as SOA. We refer to the people or entities involved in performing different stages of a workflow as *agents*. Ideally, the organization should capture all the details of the execution which includes individual agent performance, the quality of interaction among agents etc. However, issues like confidentiality, lack of visibility and in some cases lack of control, e.g., presence of Independent Service Vendors (ISVs) in workflow execution, makes it improbable (if not impossible) to capture the details at such micro level. Typically, the organizations can track only macro level features like success/failure, cost etc of a workflow. Merely aggregating the macro-level measurements offers limited insight into the individual capabilities of the agents and the effectiveness of their interaction. *In this work, we consider the issue of discovering individual traits of agents based on measurements at the level of workflows. Specifically, our goal is to develop techniques that give better insight than those obtained by simple aggregating across all workflows.* We begin by considering some examples where analysis of such macro level data can prove to be useful.

**1.1 Examples** Let us consider the case of an orchestrator of supply chains. *Orchestrator* is a management literature metaphor to describe the role of a player who organizes and manages a set of activities in a network by ensuring value-creation opportunities in the system and value appropriation

\*Contact: (sameepmehta, pvinayak)@in.ibm.com

<sup>†</sup>Center for Global Logistics and Manufacturing Strategies (GLAMS), Indian School of Business, Hyderabad

<sup>‡</sup>IBM India Research Laboratory, New Delhi

<sup>§</sup>University of North Carolina, Chapel Hill

<sup>¶</sup>The author was with IBM India Research Lab when the work was done.

mechanisms for each player. Examples of orchestrators are Li & Fung (Hong Kong) in retail [6] and Medion (Germany) in personal computing [18]. Let us consider the case of Li&Fung, which orchestrates an innovative service supply chain in apparel industry [10]. In the words of their group chairman Victor Fung, they “create a customized value chain for each customer order and orchestrate a production process that starts from raw materials all the way to the finished products”. In other words, they seamlessly bring together the large apparel retailers and a global pool of apparel-related suppliers and create value without owning any of the resources used in the production. The key differentiator that Li&Fung does bring in is the intimate knowledge of the global service network. Every major order of a retailer gets executed by creating a customized supply chain in the global network of suppliers. Each order typically gets executed by multiple workflows which go through the complete product cycle and is orchestrated by the domain experts at Li&Fung. Just the revenue of Li&Fung for the services offered is USD 11 Billion and the actual value of the retail good is greater by an order of magnitude. Identifying the providers for a particular order is based on network constraints and local economic, political, public policy, and cost considerations. But, note that each workflow is a coordinated production activity involving several independent vendors of different functionality. Therefore, the effectiveness of their execution is highly dependent on their interactions and in turn should affect how future workflows are composed. Studying the past orders, in terms of the vendors involved and the overall effectiveness of their delivery can potentially provide extremely relevant feedback about their performance in such a loosely coordinated setting.

Let us now consider the case of distributed software development. A typical software service firm delivers its services from geographically distributed locations. Teams at each location possess expertise in different functionalities from a broad domain. Examples of the domains could be product design and architecture, development capability in different technical disciplines, integration, testing, and debugging. Typical projects flow through different functionality modules of these domains at various stages of the project in a precedence oriented fashion. Parameters associated with a project such as, overheads incurred, efficiency, success etc. depend heavily on the behavioral aspects of the different teams in distributed locations and effectiveness of communication/coordination between related modules. Typically, the project manager ensures that the teams have the required skill set to *successfully* complete the project. However, choosing team with the sole intention of matching skill set demand may not produce optimal teams. In this scenario, we can view each project as a workflow and each team as an agent as it is responsible for basic units of the workflows. Organizational constraints allow us to measure the effective-

ness of each project only at a macro level rather than at the level of how individual teams performed at various stages. A firm can, not only gain better understanding of the team dynamics, but also arrive at better team compositions in future by analyzing the macro level data corresponding to the agents and the workflows in innovative ways. Recently, several research papers in the area of software engineering have considered these and other related issues in distributed software development [21, 20, 22].

**1.2 Salient Features** At this stage, we would like to highlight some of the salient features of the examples that we considered. In the case of Li&Fung in which the orchestration is over thousands of suppliers distributed globally, the team compositions are not often repeated unlike production environments where the team compositions repeat on a routine basis. The duration of each interaction is also quite short in comparison to those in production environments. Therefore, it is essential to take into account the effect of each workflow on the effectiveness of the overall service composition and delivery. We also observe that micro-monitoring of individual agents is not possible due to various reasons. In case of Li&Fung, the agents are independent entities and hence very difficult to control and monitor at a micro-level. In case of an intra-organization service chain like the software development scenario, micro-monitoring of aspects like effectiveness of communication across two dependent stages of the project could affect the team morale and also increase cost of project management. Therefore, we assume that only macro-level assessment of each workflow is available. Each workflow could be designated as having met or not met certain acceptable criteria or there could be a gradation between 0 and 1.

**1.3 Contrast with related literature** If we view the agents as entities in a network and each workflow as a relation between entities, the service delivery scenarios look similar to the traditional social networks. We briefly highlight the uniqueness of service delivery scenarios as compared to social network analysis (SNA). Traditionally SNA has on studying networks that arise by social interactions of people, socio-technological networks like WWW, and collaboration networks of scholarly articles. It has mainly concentrated on dyadic relations (sometimes even reducing relations involving multiple entities to multiple dyadic relations) and discovering structural properties of such networks. Firstly, from the point of view of service delivery, the relations are either hyper-edges over the agents or workflows involving multiple agents. Secondly, the focus of the analysis in service delivery setting is on the fact that, with each workflow, a measure of its effectiveness is recorded. The idea is to use the pattern of these interactions and their effectiveness to infer micro-level attributes of agents and their interactions.

The current problem domain is also significantly different from traditional supply chain processes. What sets services and industrial/production processes apart are (i) the degree to which human interactions dominate the total time and (ii) the degree to which human interactions influence the overall quality. To understand these differences in detail, especially from the supply chain viewpoint, we point readers to an excellent survey article by Dietrich and Harrison [8]. They highlight the novelty of the mathematical techniques required to analyze service delivery.

**1.4 Our contribution** Informally, *service interaction networks* are networks arising in service delivery, out of agents and their interactions in executing workflows. As mentioned earlier, the interaction networks provide a generalized framework to model a wide range of scenarios originating in service delivery setting. In this paper, we initiate a systematic study of this topic.

Unlike the well known sources of data for SNA such as WWW, DB LP etc., data on service interaction networks is hard to obtain. The main reason is the proprietary nature of the data. It contains crucial private data of organizations that cannot be shared externally. Anonymization of the data is not a solution either. It would reveal confidential information about the organization. For example, a company may not want to reveal that in its private estimate a certain percentage of its interactions were unsatisfactory. So, realistic data generation is a major challenge. Note that generating data by representing simplistic mathematical models for agent behavior does not work. This is due to the dominance of human aspects in service delivery. At the same time, randomized assignment of performance levels for each interaction is not helpful either. It not only results in highly inconsistent data, but is also not reflective of real-life scenarios. One would like to generate data that is close to real-life and reasonable level of consistency. Generating such a data is a research topic by itself. Towards this goal, we have developed a simulation framework. The framework can be instantiated to generate data which reflect various real life scenarios.

Related literature in SNA by computer scientists as well as sociologists offer starting points for studying service interaction networks. Particularly, the eigen vector based approaches seem to be relevant from the point of view of service interaction networks. We present several ways of extending this line of work for analyzing service interaction networks. We discuss the advantages and disadvantages of each approach.

We also develop an intuitive iterative refinement technique to compute reputations of agents that overcomes some of the limitations of simply using the aggregates. We illustrate the efficacy of our approach both conceptually and empirically on data generated by the simulation framework. We also consider the impact of specific “links” on the service de-

livery. For example, agents  $a$  and  $b$  might be effective agents individually. But, workflows in which a task performed by  $a$  is followed by  $b$  might not be as effective. This suggests that the collaboration between  $a$  and  $b$  is not as desirable and should be avoided in future workflow compositions. Our work introduces a very important analysis domain for service delivery and presents several interesting future directions.

## 2 Analysis Framework

In this section, we formulate the problems considered in this paper. For ease of exposition, we call an organization responsible for end-to-end service delivery as the orchestrator. We call entities (be it individuals, teams or independent service providers) that are responsible for independent units of work as agents.

**2.1 Service Interaction Networks** In our setting there is an orchestrator  $\mathcal{O}$  who can avail the services of a set of agents  $\mathcal{V} = \{1, 2, \dots, N\}$ . The orchestrator  $\mathcal{O}$  specializes in accomplishing high-level tasks from a broad domain and the agents provide the different functionalities to accomplish the high-level tasks. Given a task  $t$ , the orchestrator composes a workflow consisting of a subset of  $\mathcal{V}$  to accomplish the tasks. We assume that the set  $\mathcal{T} = \{t_1, t_2, \dots, T\}$  denotes the set of tasks performed by the orchestrator over a significant period of time. In the most general form a workflow  $W_t$  can be represented as a graph  $G_t(V_t, A_t)$  with vertices  $V_t \subseteq \mathcal{V}$  and edges  $A_t$  between vertices in  $V_t$  that model the direct interactions between any two actors in  $V_t$ . However, certain tasks require repeated interactions which are cumbersome to represent in terms of direct edges. Therefore, we assume that each task is accomplished by a workflow which is represented as a hyper-edge connecting all the involved agents. Abstractly, the hyper-edge highlights the fact that an interaction involving its agents took place before the task could be accomplished. Associated with each hyper-edge is an “outcome” which is a measure of the effectiveness of its execution. In general, the outcome could be a real number between 0 and 1. But, our main interest will be in the special case when the workflows are either “successful” (i.e, 1) or “failure” (i.e, 0). We call the collection of agents along with the hyper-edges representing the various tasks and their outcomes as *service interaction networks*. An instance of a service interaction network is given by  $(\mathcal{V}, \mathcal{T}, \cup_{t \in \mathcal{T}} (W_t, R_t))$  where the tuples  $(W_t, R_t)$  associate workflows with their outcomes. Figure 1 shows an example with  $N = 7$  and  $T = 5$ .

Typically, workflows are composed by the orchestrator based on (i) system constraints in terms of capacity and capabilities of the agents, and (ii) social, geographical, political, and economic factors. In addition, the orchestrator has to take into account the individual and team dynamics of the agents. As explained in the examples presented in the intro-

duction, it is not always feasible to micro-monitor the workflow executions. So, one of the main themes of analyzing service interaction networks is to infer useful information about the agents from the high-level measurements across workflows. For example,

- How does an agent impact the success of the workflows in which it is involved?
- What is the impact of involving a specific subset of agents on an workflow? Specifically, how does the existence of a link between two agents affect the success of workflows in which they are involved?
- Are there any dominant or influential agents emerging in the network?

Note that an obvious approach to tackle the above questions is based on aggregating. For example, to measure the effectiveness of an agent, one could just measure the fraction of the successful workflows of all the workflows in which the agent is involved. Similarly, we can extend it to assess effectiveness of direct links and larger subsets. Some of the limitations of such an approach are noted below.

Consider a successful and influential *super-agent*  $a$  who ensures that all the workflows that he is involved in succeed. Consider an agent  $b$  who is *failure-prone* when teamed with ordinary agents. Let 80% of the workflows of  $b$  also contain  $a$ . Let us further say that a large fraction of the rest of  $b$ 's workflows have failed. The aggregate based approach would infer that  $b$  is a highly successful agent. However, we would like to infer that  $b$  is prone to failures (and may be needs to be targeted in training programs). A different problem arises when we use aggregates to make transitive inferences. Suppose  $c$  and  $d$  are indeed effective agents. An aggregate based method would suggest that it is okay to compose a workflow involving  $c$  and  $d$ . However, a closer look at direct interactions between  $c$  and  $d$  may suggest otherwise.

We consider algorithmic approaches to tackle the above questions. We would like them to deal with behavioral inconsistencies of agents, a commonly encountered phenomena in service delivery settings.

**2.2 Related Literature** In this section, we will summarize related literature and highlight the uniqueness of our formulations with respect to previous work. Some of the previous works need to be examined more closely to ascertain their applicability to our formulations. Section 3 considers some of the previous works as starting point for representation and analyses. The purpose here is to briefly highlight the uniqueness of our work.

The distinction of our setting with the traditional SNA focusing on structural properties of networks of dyadic relations was highlighted in the introduction. An excellent survey of the structural analyses in SNA can be found in articles

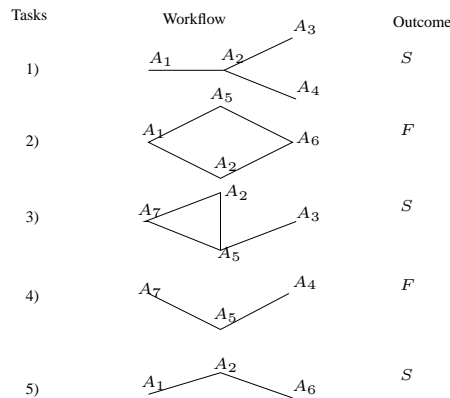


Figure 1: An example with  $N = 7$  actors and  $T = 5$  workflows

by Newmann [15, 16, 17]. Some of the well known examples are the studies on the “small world” phenomena in such networks [13] and studies on the “shape” of the WWW [5]. Most studies in SNA do not focus on the properties of the links themselves. Recently, in the context of analyzing telephone call-graphs [14, 19], the weights on the links was used to discover communities. However, in our case, the label on the hyper-edge is indicative of the effectiveness of the corresponding workflow. Moreover, we are interested in inferring reputations of individual agents and their interactions.

If we denote the agents as nodes and the interactions between them as edges, then we have interaction networks, much similar to social networks. *Centrality* concepts are widely used in social networks for studying structural importance of nodes in the network. For interaction networks, this translates to quantifying the status of the agents based on their interactions. However, application of centrality for interactions networks is non-trivial due to the presence of outcome of each workflow and there exists preferential relations over the outcomes ( $S$  is preferred over  $F$ ). In the next section, we show how to construct an interaction network from the workflows and how centrality concepts can be applied to interaction networks.

### 3 Interaction Network as Graphs

In this section, we model the interaction networks as social networks. The basic mathematical structure for visualizing the social network is a graph. The agents are represented by nodes with the linkages between the agents as edges. The linkages could arise due to evaluation of one person by another (friendship, liking, respect), transfer of material resources (business transactions, lending or borrowing things), association or affiliation (club membership), behavioral interaction (talking together, sending messages), formal relations such as authority, and biological relationships such as

kinship or descent. A linkage establishes a tie at the most basic level between a pair of agents. The tie is an inherent property of the pair. The linkage in a interaction network is the quantification of the past interactions between any two agents that happened in the workflows.

Social network analysis is concerned with understanding the linkages among agents and the implications of these linkages. Some of the popular topics of focus are: (1) quantifying the structural importance of agents in the network; (2) identifying the core and peripheral agents; (3) analyzing groups and sub-groups; and (4) structural measures of social capital to identify what features of network contribute to agents. In this work, our focus is on identifying key agents based on the past interactions with the other agents, working as teams. If the interaction network is modeled as a graph with agents as nodes, then the edges characterize the strength of the interaction. The *key* agents can then be identified by their structural importance in the network.

A relevant stream of research in social network analysis that measures the structural importance of an agent relative to that of the others in the network is *centrality*. There are numerous standard measures of centrality [23], each appropriate for different environments. The four popular centrality measures are *degree*, *closeness*, *betweenness*, and *eigenvector* [9, 2].

- *Degree*: Centrality of an agent is a function of number of other agents to which it is adjacent (or directly connected).
- *Closeness*: An agent that can be easily reached from other agents is more central.
- *Betweenness*: An agent that lies on more number of geodesic paths (shortest paths) between any two agents in the network has a higher centrality score.
- *Eigenvector*: The centrality of an agents depends on the centrality of the agents to which it is adjacent.

The closeness and betweenness measures are appropriate for networks that model flows and transmissions. Degree centrality is concerned with *how* many others an agent is linked with. For interaction networks, the centrality of an agent depends on *whom* he interacts with and *how many* of them. Eigenvector centrality [2] is the appropriate measure for the interaction networks. It takes into account the centralities of the other agents with which an agent interacts. This is relevant to our objective of studying the impact that individuals have in team setting.

Firstly, the interaction network has to be modeled as a graph from the workflows. A distinct feature of our problem is the presence of outcomes for the workflows, which have to be taken into consideration. Given that the binary outcomes  $S$  and  $F$  are diametrically opposite in their

values, there are two natural approaches: (1) to construct two different interaction networks, one for each outcome, and (2) to construct a signed value graph with positive edges to denote interactions of  $S$  workflows and negative edges for  $F$  workflows. First, we shall adopt both these approaches and discuss their limitations. Then we propose a novel approach that can also model workflows with finite discrete outcomes, rather than just  $S$  and  $F$ .

Let  $\Delta = [\delta_{ij}]$  be the  $N \times N$  *interaction* matrix that captures the strength of interactions between the  $N$  agents. If  $(i, j)$  is the edge between  $i$  and  $j$  in the interaction network, then  $\delta_{ij}$  is the weight on that edge. The  $\Delta$  has to be constructed from the workflows and let us ignore the outcomes for brevity. With a slight abuse of notation, let  $\delta_{ij}^t$  denote the *strength* of interaction between actors  $i$  and  $j$  in workflow  $t$ . Following are some of the desirable properties of  $\delta_{ij}^t$ .

1. **[No influence]** If  $i \notin V_t$  and  $j \notin V_t$ , then  $\delta_{ij}^t = 0$ .
2. **[Intra-workflow influence]**  $\delta_{ij}^t$  should be function of the path length between  $i$  and  $j$  in  $G_t$ .
3. **[Asymmetry]** Based on the relative influence of the roles of  $i$  and  $j$ ,  $\delta_{ij}^t \neq \delta_{ji}^t$ .
4. **[Past influence]** The overall interaction  $\delta_{ij}$  can be derived as:

$$\delta_{ij} = \sum_t \mu^t \delta_{ij}^t, \mu^t \leq \mu^{t+1}$$

The first property is a trivial one that emphasizes the importance of participation in the same workflow as the necessity to qualify the linkage as interaction. The *intra-workflow influence* allows to differentiate intra-workflow and inter-workflow interactions. Let  $\hat{t}$  and  $\tilde{t}$  be two different workflows. Let

$$(3.1) \quad (i, j), (j, k) \in E_{\hat{t}} \quad (i, k) \notin E_{\hat{t}}$$

$$(3.2) \quad (j, l) \in E_{\tilde{t}} \quad i \notin V_{\tilde{t}}$$

The traditional assignment of weights in social networks emphasizes only on direct interactions:

$$(3.3) \quad \delta_{ij}^{\hat{t}} = \delta_{jk}^{\hat{t}} = 1; \quad \delta_{ik}^{\hat{t}} = 0;$$

$$(3.4) \quad \delta_{jl}^{\tilde{t}} = 1; \quad \delta_{il}^{\tilde{t}} = 0;$$

Intuitively,  $\delta_{ik}^{\hat{t}} > \delta_{il}^{\tilde{t}}$ , as  $i$  and  $k$  have worked on the same project, whereas  $i$  and  $l$  have not. Thus the intra-workflow influence property is desirable if we have to differentiate above scenarios. The asymmetry property is useful if the responsibilities of  $i$  and  $j$  are hierarchically structured. If  $i$  is above  $j$ , then influence from  $i$  to  $j$  will be more than that of the reverse. The last property is about how

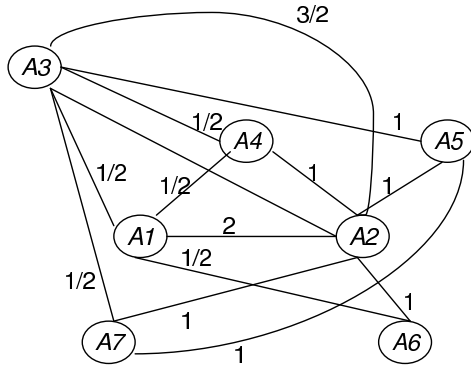


Figure 2: Interaction network constructed from workflows with outcome  $S$

we obtain the overall strength of interaction based on the individual workflow interactions. The past influence factor  $\mu^t$  can be used to model the relative importance of the influence with respect to time. One can easily see that by judiciously choosing  $\mu^t$ , we can arrive at various kinds of past influences: *Uniform, sliding window*, etc.

We will now explore the approach of modeling the outcomes using two different interaction networks. Social networks allow multiple relations to be defined on the same set of agents. Hence, one can define linkages related to  $S$  and that of related to  $F$  workflows. For brevity, we consider them as two different networks with interaction matrices  $\Delta^S$  and  $\Delta^F$ . We will use the example in Figure 1 to illustrate this approach. The  $\Delta^S$  is constructed only from  $S$  workflows and we assume symmetry  $\delta_{ij}^t = \delta_{ji}^t$ . The intra-workflow influence is modeled as follows:

$$(3.5) \quad \delta_{ij}^t = \frac{1}{P_{ij}^t}$$

The  $P_{ij}^t$  is the shortest path length between  $i$  and  $j$  in workflow  $t$ , where each edge has unit weight. As the example has few workflows, the past influence is *uniform*:  $\mu^t = 1$ .

Figure 2 shows the interaction network of agents modeled from the workflows with  $S$  outcomes in Figure 1. Let  $x_j$  denote the eigenvector centrality of agent  $j$  and  $\Delta^S$  denote the interaction matrix. The eigenvector centrality [2] is given by:

$$(3.6) \quad \Delta^S x = \lambda x$$

The  $\lambda$  is the largest eigenvalue of  $\Delta^S$  and  $x$  is its corresponding eigenvector. In the linear summation form, the above reduces to:

$$(3.7) \quad x_j = \frac{1}{\lambda} \sum_i \delta_{ij}^S x_i, \quad \forall j$$

Eigenvector centrality measures the centrality of an agent as a linear combination of centralities of agents to which it is connected. Unlike degree, which weighs every adjacent agent equally, the eigenvector weights adjacent agents according to their centralities. The agents can then be ranked based on the component wise value of the eigenvector: If  $x_{j1} \geq x_{j2} \geq x_{j3} \dots$ , then the ranking for the agents are  $j1, j2, j3, \dots$ . The idea of using the eigenvector to do ranking dates back to the 1950's [24, 11], which was also later used to rank pages in the web.

One can similarly construct an interaction graph for workflows with  $F$  outcomes. The rankings for the interaction matrices  $\Delta^S$  and  $\Delta^F$  are:

$$\Delta^S : A2, A3, A1, A5, A7, A4, A6$$

$$\Delta^F : A5, A1, A6, A2, A7, A4, A3$$

The above rankings are of little help as one cannot be used ignoring the other and both together cannot yield consistent ranking. For example,  $A1$  is preferable to  $A5$ , as  $A1$  is ranked higher than  $A5$  in  $S$  ranking and vice versa in  $F$  ranking. However, the preference between  $A1$  and  $A4$  cannot be deduced. Hence, even at the level of pairwise comparisons, it is not possible to derive preferential rankings.

The second approach is to model the interaction network as a signed graph with both positive and negative links. The signed interaction matrix is given by:

$$(3.8) \quad \Delta^S - \Delta^F$$

For handling relations with negative status, eigen vector based centrality measure was proposed in [4]. However, it is only applicable to matrices with *balanced structure*. If positive edges denote friendship and the negative ones denote enmity, then the balance structure exists if friend of friend is a friend, enemy of a friend is an enemy, friend of an enemy is an enemy, and enemy of an enemy is a friend. For the signed graphs with balanced structures, the vertices can be partitioned into two subgraphs such that the links within each subgraph is positive and the links between the two graphs is negative. One can extend beyond balanced structures and consider clusters, with more than two subgraphs. For our problem, clusters and balance structures are not applicable as the pairwise interactions between any two agents cannot be claimed to be positive or negative based on the outcomes. In the example in Figure 1,  $A2$  and  $A6$  have directly interacted in one workflow with  $S$  outcome and one with  $F$  outcome. Hence, one also directly cannot use techniques for ranking webpages with negative links.

The problem with the above modeling approaches is the implicit assumption that  $S$  and  $F$  are diametrically opposite outcomes. In the following, we assume them to be outcomes with different *utilities*, including negative. Then the pairwise

interactions cannot be negative by themselves, even if it has resulted in an  $F$  outcome. We propose a novel technique, where the outcomes  $\{S, F\}$  are included as agents in a directed network. The main rationale for the specific directed graph construction that we use is the fact that the outcomes have certain fixed utilities and we would like to rank the agents based on the utilities of the outcomes; that is, there is inherent asymmetry between the normal agents and the special agents corresponding to  $S$  and  $F$ . Therefore, we create directed edges that allow the utilities (or the influence) of the outcomes to be transmitted to the agents, but disallow any transmission in the reverse direction. Details of the construction are as below:

$$(3.9) \quad \delta_{S,j}^t = \begin{cases} 1 & \text{if } R_t = S \\ 0 & \text{otherwise} \end{cases}$$

$$(3.10) \quad \delta_{F,j}^t = \begin{cases} 1 & \text{if } R_t = F \\ 0 & \text{otherwise} \end{cases}$$

The interaction between the outcomes and the agents are directed:

$$(3.11) \quad \delta_{j,R_t}^t = 0, \forall j, \forall t, \forall R_t$$

The interaction matrix  $\Delta$  is of order  $(N + 2) \times (N + 2)$  that is directed and hence not symmetric:

$$(3.12) \quad \delta_{ij} = \delta_{ij}(S) + \delta_{ij}(F), \forall i, j \in \mathcal{V}$$

$$(3.13) \quad \delta_{R_t,j} = \sum_t \delta_{R_t,j}^t, \forall j, \forall R_t$$

$$(3.14) \quad \delta_{j,R_t} = \sum_t \delta_{j,R_t}^t, \forall j, \forall R_t$$

The directed interaction network corresponding to the example in Figure 1 is shown in Figure 3.

One of the problems with the directed construction is that the eigenvalue centrality cannot be directly used for directed graphs. However, we leverage a result by Bonacich [3] in which it is shown that a centrality measure called *alpha*-centrality (similar in spirit to the eigenvector centrality) can be computed for directed graphs. The main computational task of the *alpha*-centrality is given by the following:

$$(3.15) \quad x = (I - \alpha \Delta^T)^{-1} e$$

The vector  $e$  is exogenous source of information and parameter  $\alpha$  reflects the relative importance of endogenous ( $\Delta$ ) versus exogenous ( $e$ ) factors in the determination of centrality. The centrality  $x$  is shown to exist for  $0 \leq x < 1/\lambda$ , where  $\lambda$  is the largest eigenvalue of  $\Delta$ . The  $e$  is an external source of status that is independent of the interactions. Let  $y_{ij}$  denote the entries in the inverse matrix  $(I - \alpha \Delta^T)^{-1}$ . Then the

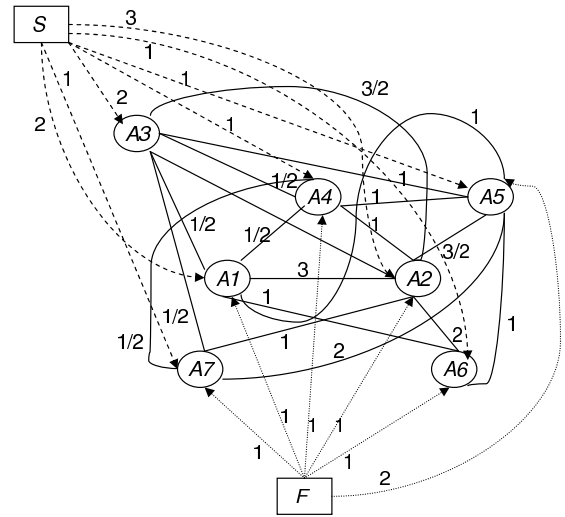


Figure 3: Interaction network with  $S$  and  $F$  as agents

centrality scores are given by:

$$(3.16) \quad x_j = \sum_i y_{ji} e_i$$

Note that by construction  $y_{R_t,i} = 0$ ,  $y_{ii} = 1$ ,  $\forall i$  and  $\forall R_t$ . Hence, the centralities of the *outcomes* are unaffected:  $x_{R_t} = e_{R_t}$ . If the vector  $e$  denotes the judiciously chosen utilities, then the centrality score essentially captures both the interactions among agents and outcomes, and the utilities of the outcomes. Following is one possible assignment to  $e$ :

$$(3.17) \quad e_j = 1, \forall j \in \mathcal{V}$$

$$(3.18) \quad e_S = -e_F$$

All the agents have equal status and the  $S$  and  $F$  have opposite status with some high value other than 1. Table 1 shows the rankings obtained for  $e_S = 10$ ,  $e_F = -10$ , and for various values of  $\alpha$ . The largest eigenvalue of  $\Delta$  is  $\lambda = 6.222$ . Observe that in a suitable range of  $\alpha$ , the rankings seem to show consistency. As far as we know, there is no study on choosing the appropriate  $\alpha$  and  $e$  that gives rise to consistent rankings. We are currently exploring ways to automatically compute the range for  $\alpha$  and the vector  $e$  that give rise to consistent rankings.

The proposed technique of modeling  $S$  and  $F$  as agents can easily be extended to finite discrete outcomes with different utilities. This section dealt with ranking of agents based on historical interactions and outcomes. In the next section, we develop an iterative algorithm that aims to discover how the agents contribute to the outcomes of their workflows.

$\alpha$	Ranking
0.01	A2, A3, A1, A4, A6, A7, A5
0.02	A2, A3, A1, A6, A7, A4, A5
0.05	A2, A3, A1, A6, A7, A4, A5
0.08	A2, A3, A1, A6, A5, A7, A4
0.10	A2, A1, A3, A6, A5, A7, A4
0.12	A2, A1, A3, A5, A6, A7, A4
0.15	A2, A1, A5, A3, A6, A7, A4

$e_j = 1, \forall j; e_S = 10; e_F = -10; \lambda = 6.222;$

Table 1: Ranking of agents using  $\alpha$ -centrality measure

#### 4 Iterative updates approach

In this section, we present an iterative approach to infer useful information regarding the impact of agents on their workflows.

One of the ways of measuring the impact of an agent is to assign a weight in the range of  $[0, 1]$  that is indicative of the agent's contribution towards success/failure of workflows. Let  $w_a, \forall a \in \mathcal{V}$  be the assignment of weights to the agents. Given these weight assignments and a specific workflow  $W$ , let  $w_{avg} = \frac{\sum_{a \in W} w_a}{|a \in W|}$  be the average weight of the agents belonging to the workflow. One way to explain the outcome of the workflow is to compare the average weight to certain thresholds associated with the outcomes. For instance, let  $S_t$  and  $F_t$  be two thresholds in the range  $[0, 1]$  corresponding to successful and failure workflows respectively. The assignment of weights is said to explain the outcome of a successful workflow  $W$  if  $w_{avg} > S_t$ . Similarly, for a failed workflow, we require  $w_{avg} < F_t$ .

For simplicity, let us assume that the outcomes  $R_t$ s are either 0 (failure) or 1 (success). Our approach extends directly to the continuous case. Aggregate based method of assigning weights would average the outcomes of the agent's workflows. For an agent  $a$ ,  $w_a = \frac{\sum_{t: a \in W_t} R_t}{|\{t: a \in W_t\}|}$ . Let  $f$  be the fraction of workflows that are explained by the aggregation based approach. Our goal is to significantly improve the fraction of explained workflows in comparison to  $f$ .

Our idea is very simple. We start with any valid assignment of weights to agents. For each workflow that is not explained by the current assignment, update the weights of the agents belonging to the workflow in small quantities in such a way that the gap between the threshold and its average decreases. Specifically, we iteratively improve the assignment by considering all unexplained workflows in each iteration. For each workflow that is not explained, we update the weights of all the agents involved in its execution by a very small quantity  $\epsilon$ . If a workflow's outcome is 1 and it is not explained, we increment the weight of each of its agents by  $\epsilon$ . Similarly, if the workflow's outcome is 0 and it is unexplained, then, we reduce the weights of each of its agents by the same quantity  $\epsilon$ . While updating the weights

in this fashion, we restrict them to the  $[0, 1]$  range. At all time, we maintain the best assignment so far. The procedure is terminated when the fraction of the explained workflows is above a threshold  $F$  (say 0.95) or if the fraction of the explained workflows by the assignments in the last  $L$  rounds does not increase by a minimum threshold,  $M$ . As we can always begin with aggregate weights,

**PROPOSITION 4.1.** *The assignment returned by the iterative approach is at least as good as the aggregate based method.*

Our algorithm is also very fast. It is easy to verify the following:

**PROPOSITION 4.2.** *The running time of the iterative approach is  $O(\frac{L}{M} \cdot |I|)$  where  $|I|$  represents the size of the input.*

Our approach is easily extended to infer the impact of the interactions as follows. Each direct interaction or a link is modeled as a *link agent* with a weight associated with it. The algorithm just presented can now be run on the set of individual agents and link agents with the outcomes of the workflows as the input. In fact, when the workflow is a precedence relationship, it does not even change the running time. This approach can be extended to larger subsets in a limited way. For subsets of constant size, say  $k$ , we can consider *hyper-edge agents* of size  $k$ . In this case of course, the running time would also depend on  $N^k$ .

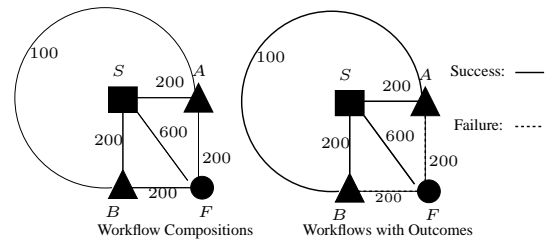


Figure 4: An example that shows the advantage of iterative approach.

To get an understanding of how our approach helps us get better explanations, consider the example shown in Figure 4. In this example, there are four agents, namely,  $A, B, F, S$ . The LHS of the figure shows how a set of 1500 workflows are assigned as edges between two agents. The numbers on the edges indicate the number of tasks assigned to the edge. For example, 100 tasks are executed by an interaction between  $A$  and  $B$ . The RHS of the figure shows the outcomes of the workflow executions. The successful workflows are highlighted by solid lines and the unsuccessful ones by dotted lines. Agent  $S$  is the most effective agent who is able to ensure success of all the workflows he is assigned. A naive aggregation approach

would assign effectiveness values of 1.0, 0.6, 0.6, and 0.6 to  $S, F, A, B$  respectively. If we assume that an average above 0.5 is required to explain success and an average below 0.5 is required to explain failure, then, these values are able to explain only 1100 workflows. The failed workflows between  $F, A$  and  $F, B$  are not explainable. However, a single iteration of our refinement algorithm with  $\epsilon = 0.0004$  computes the effectiveness of different agents as:  $e_S = 1, e_F = 0.34, e_A = 0.52, e_B = 0.52$ . It is easy to check that these values explain all the workflows, including the failed ones. Thus, our heuristic is designed to capture this process of readjustment by focusing on workflows that need explanation. We present the details of our experimentation with this approach in the next section.

**Discussion.** Note that the weights of  $A$  and  $B$  in the above example is not a good reflection of their abilities. However, when these weights are used in future assignments, their true ability will be reflected. Readers familiar with the multiplicative weights update methods [12, 1] would immediately notice that our updates are additive. The reason is that our notion of explanation is based on averages. For instance, in the above example, penalizing unexplained failed workflows by a multiplicative update of  $(1 - \delta)$  would drive the weights of  $A, B, F$  to very small values, thus worsening the solution. Perhaps, one could change the notion of explanations to suit the multiplicative updates as well. Here, we restrict ourselves to average based explanations and hence additive updates.

## 5 Computational Results

As mentioned in the introduction, it is difficult to get real-life data on the effectiveness of the workflow executions in service delivery. At the same time, modeling agents and interactions by standard mathematical models is not useful either as they are not representative of real-life data. Therefore, it is essential to generate data that is as close to the real-life data as possible. In context of service delivery where there is a major fraction of human element involved, the task of generating realistic data is a challenging research topic in itself. Techniques for such a simulation framework is not the main focus of this paper. However, developing a simulation framework and techniques to generate realistic data is one of our long-term goals. Here, we present the current implementation of our simulation platform and how we use it for our experimental purposes. Our framework is flexible and easily extensible so that experts with domain knowledge can quickly transform their experience to usable models that can be used by the simulator.

**5.1 Simulation Platform** The goal of the simulation platform is to build a simulation environment consisting of i) a set of agents each of whom is capable of performing a specified set of activities, ii) to generate realistic workflows

requiring different functionalities in the form of a chain (to begin with), iii) to push the workflows through the network of agents in which the agent behavior is close to real-life, and (iv) to measure the outcome of these workflow executions. In the order of their simplicity in implementation, this involves three aspects (i) mechanism to create the network of agents and the set of workflows (ii) modeling the behaviors of the agents in the face of a sequence of workflow assignments (iii) ways in which the workflows are assigned to the agents during the simulation. We present the details of our platform in this sequence, simple to complex.

The simplest part of the simulation framework is to instantiate it by creating the agents and setting the capacities of the agents. In our formulation, we mentioned that the workflows can be general subgraphs on the set of agents. However, currently in our framework, we represent workflows as precedence driven chains. Each stage of the chain is specified by a functionality required at that stage. Each agent is endowed with a capacity and a set of functionalities that it can perform. So, the assignment of workflows to agents has to satisfy the following constraints: (i) each stage is assigned to an agent who can perform the required functionality and (ii) the capacity constraints of the agents and the links assigned to the workflow are not violated. There are also capacities on the link between two agents that capture the operational constraints of interaction between the two agents. Our simulation framework allows simple text based initialization of these fields from domain experts. It can also be initialized based on certain well observed phenomena like power law etc. When a workflow execution is completed, the corresponding agents recover the capacities reserved for it. All along, we associate a weight  $q_a$  with each agent  $a \in \mathcal{V}$  which is independent of the weights assigned by the algorithm in Section 4. These weights are used in the simulation.

Modeling the behavior of the agents is a very key aspect of the simulation. Clearly, it is unreasonable to capture the human element by simplistic mathematical models. Various parameters, such as external factors, personal traits of agents etc. can affect their behavior and are difficult to capture mathematically. Two different effects of agent behavior on the service delivery should be captured as well. Firstly, even competent agents can feel the monotonicity of similar jobs and might slip into complacency, especially when performing similar tasks on a repeated basis. Secondly, the agents also have a capability to learn on the job thus making them decidedly different from machines.

We model the complex patterns of agent behavior by starting with simple models and modifying such predictable behavior with unpredictable and dynamic aspects. The simple form of an agent's behavior is a normal distribution  $N(p, v)$  where  $0 \leq p \leq 1$  is the mean of the weights it contributes to its workflows,  $v$  is the variance of the contribution. Associated with each agent is a parameter

$d, 0 \leq d \leq 1$  that indicates the probability that the agent's performance deviates from expected behavior and takes a random form. This captures the unpredictability aspect of an agent.

We extend this simple framework to incorporate different aspects of agent behavior as follows. Firstly, we treat the succession of similar jobs done by an agent as a random walk with success probability corresponding to its standard profile. Now, the complacency aspect is introduced by asking the question, "what is the likelihood of a long walk of only successes in such a random walk?" Based on this probability, we introduce complacency at a given stage for a given agent. The learning aspect is handled as follows. Let us consider a workflow that is assigned to a subset of agents  $\mathcal{A}$ . Let the average weight of the agents as maintained by the simulation engine is above  $S_t$ . We treat the workflow as a success. Even though the workflow was a success, some of the agents may have a performance model which indicates an expected response below  $S_t$ . We take this as an instance of the agent learning on the job. Once an agent gets ticked a certain number of times, we update his performance model to capture this learning. Lastly, to simulate scenarios such as the one illustrated in Section 4, we also add special agents characterized as *super-agents* and *failure-prone* agents. The failure prone agents are superseded by the presence of super-agents.

The trickiest part of the simulation is the assignment of the workflows to the agents. The assignments affect the agent behavior in a non-trivial manner as described above. In our current implementation, we have multiple heuristics for assigning workflows to the agents and we generate data corresponding to each of the options. One of the ways of assigning workflows is to "load balance" the agents in terms of their capacities. Second is the greedy way: choose that set of agents which has the required capacity and has best expected weight of success. Note that this aspect affects the complacency of the agents and we do not have any means of controlling it as of now. The third option is to make random assignment at each stage of the workflows: among all the agents with surplus capacity and who can perform the required functionality, choose one randomly.

**5.2 Experimental Evaluation** We used the simulation framework to carry out our experiments. We generated instances of service interaction networks by first creating a network of agents and then pushing a certain number of workflows through them. The outcome of the workflows was decided by the average of the weights of the assigned agents ( $q_a$ s maintained by the simulation framework). The iterative analysis presented in Section 4 was run on the resulting service interaction networks. The efficacy of our approach was measured based on the how it could improve the fraction of the workflows that were explained by the final weight assignment as opposed to the fraction explained by the simple

aggregation method.

Let  $T$  be the number of workflows and  $N$  be the number of actors in the service interaction network. We found that the improvement achieved by the iterative technique is not uniform for all combinations of  $T$  and  $N$ . Our experiments suggest that in a particular range of the ratio of number of tasks to the number of actors,  $T/N$ , the iterative algorithm seems to do particularly well. We also studied the effect of the presence of super-agents and failure-prone agents on the performance of the iterative technique. We generated instance with varying number of workflows and agents, with and without the presence of special agents. For assigning the workflows to the agents, we followed all the three policies mentioned in the Section 5.1. In our set-up, we used  $S_t = F_t = 0.5, \epsilon = 0.005$  and different values of the threshold  $F$ , the fraction of workflows that need to be explained for the algorithm to terminate. A summary of the our findings is:

- The performance of the iterative approach is not affected significantly by the presence of special agents.
- The performance of the iterative approach is not affected significantly by the choice of the heuristic for assigning workflows to the agents. The improvement is slightly less in the case of greedy assignment.
- In general we are able to improve the fraction of the explained workflows in the range of 0.20. As expected, we do special agents when they are present.

As mentioned above, the improvement achieved by the iterative method is not uniform over different  $T/N$  ratios. One of the patterns we observed is that when the ratio is close to 1, the improvement is not significant as the average based approach performs quite well. As the ratio increases, it improves upto a certain threshold. When the ratio is very high, say  $O(N)$ , i.e., number of workflows is  $O(N^2)$ , the improvement is not significant. This is due to the fact that the complexities introduced by the model make it very difficult to explain the outcomes by associating a single weight to the agents. The table in Figure 5 shows the improvement while keeping the number of actors fixed. Figure 6 shows an approximate plot of how the  $T/N$  ratio affects the performance. Note that the horizontal axis is not to scale. The purpose of the figure is to capture the trend at the two extreme sides of the input sizes.

Our experiments with different strategies for assigning the workflows to agents did not affect the performance of the approach in any significant way. It only lowered the improvement to the range of 0.15. But, the patterns of improvement over different  $T/N$  ratios remained same. Therefore, we present our experimental numbers across all our experiments without making a distinction about the workflow assignment method. The table in Figure 7 shows the performance of the two approaches for different combinations of

#Agents	#Workflows	% of Ag. Apr.	% of Iter. Apr.
250	500	74	90
250	2000	64	81
250	10000	57	77
250	60000	62	78
250	80000	46	61
125	500	67	82
125	1500	62	81
125	5000	63	79
125	10000	66	78
125	30000	44	52

Figure 5: Comparison of the two approaches as  $T$  is increased while keeping  $N$  constant

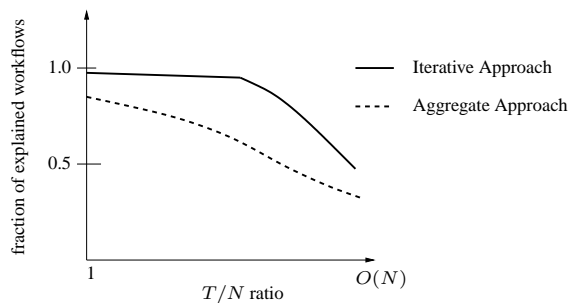


Figure 6: An approximate plot of how the algorithm performs as  $T/N$  ratio is varied.

$T$  and  $N$ . For each combination, we generated many instances and the table captures the average behavior. The mean and the variance of the improvement obtained by the iterative method over all the experiments were 0.19 and 0.05 respectively.

## 6 Discussion and Future Work

Analyzing service interaction networks is one of the important aspects of the emerging service engineering discipline. In this section, we highlight some of the interesting research problems in this area. We begin by first highlighting some of the interesting extensions to the work presented here.

One of the reasons for the dip in the performance of the algorithm as the workflows increase is our attempt to capture the entire dynamics by a single number. One of the approaches to deal with this problem is to divide the workflows temporally into multiple epochs and to maintain a separate weight for each epoch. This would improve the performance of both aggregate based method and the iterative method. Identifying the epochs automatically seems to be an interesting way to extend our work.

In our current implementation, the workflows are precedence chains. This makes it easier for the iterative approach

#Agents	#Workflows	% of Ag. Apr.	% of Iter. Apr.
6	10	77	90
10	10	73	91
100	100	69	84
125	1000	62	81
125	5000	63	79
250	500	71	92
250	5000	56	79
250	100000	38	50

Figure 7: Comparison of the two approaches for different choices of  $T$  and  $N$ .

to include link agents and get similar performance benefits. It is not difficult to extend the simulation framework to generate workflows that are subgraphs on the set of agents. But, capturing the impact of interaction between subsets of agents now becomes tricky. The iterative approach does not extend naturally. We are exploring algorithmic approaches in this generic scenario.

The eigen vector based approaches have had success in SNA and in related applications like page ranking. A recent research paper by Kerchove and Dooren [7] has extended this approach when negative links are present in the network. We are exploring the eigen vector based approach further in our current work.

Let us now highlight other interesting problems in analyzing service interaction networks. Consider the service interaction network that emerges in the example of Li&Fung. As the agents in this case are ISVs, they do not possess the knowledge of the global network. However, as different workflows are passed through them and they interact with the other ISVs, they start discovering parts of the network. When a large number of workflows have been pushed through the system, an interesting question that emerges is: *is there a small set of agents who, if they combine their local knowledge of the network, can reconstruct a significant portion of the global network?* This seems to be an important aspect of risk analysis in moderately sized networks. Even in intra-organizational networks, such information can be used to promote the identified agents to achieve better overall coordination. Modeling the problem formally and developing algorithmic approaches to this problem is a promising research direction.

As highlighted in the introduction, access to real-life data is a major hurdle for studying service interaction networks, especially from the point of view of public dissemination. Therefore, realistic data generation using novel simulation techniques and exploiting domain knowledge of experts is essential for studying service interaction network. The simulation framework presented here is only a first step in this direction. We are currently focusing on exploiting a combination of innovative modeling and integration of do-

main expertise to make our simulation framework richer and adaptable to wider range of services scenarios

## References

- [1] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: A meta algorithm and its applications. Technical report, The Princeton University.
- [2] Philip Bonacich. Factoring and weighting approaches to clique identification. *Journal of Mathematical Sociology*, 2:113–120, 1972.
- [3] Philip Bonacich and Paulette Lloyd. Eigenvector-like measures of centrality for asymmetric relations. *Social Networks*, 23:191–201, 2001.
- [4] Philip Bonacich and Paulette Lloyd. Calculating status with negative relations. *Social Networks*, 26:331–338, 2004.
- [5] Andrei Z. Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet L. Wiener. Graph structure in the web. In *Proceedings of the ninth international conference on World Wide Web (WWW)*, pages 309–320, 2000.
- [6] John Seely Brown, Scott Durchslag, and John Hagel. Loosening up: How process networks unlock the power of specialization. In *The McKinsey Quarterly: Special edition on risk and resilience*. 2002.
- [7] Cristobald de Kerchove and Paul Van Dooren. The pagetrust algorithm: How to rank web pages when negative links are allowed? In *SDM*, pages 346–352, 2008.
- [8] Brenda Dietrich and Terry Harrison. Serving the services industry. *Operations Research and Management Science (ORMS)*, 33(3), 2006.
- [9] Linton C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1:215–239, 1979.
- [10] John Hagel, Scott Durchslag, and John Seely Brown. Orchestrating loosely coupled business processes: The secret to successful collaboration. Copyright by John Hagel III, John Seely Brown, and Scott Durchslag; available at [http://www.johnhagel.com/orchestrating\\_collaboration.pdf](http://www.johnhagel.com/orchestrating_collaboration.pdf), 2002.
- [11] M. G. Kendall. Further contributions to the theory of paired comparisons. *Biometrics*, 11:43, 1995.
- [12] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.
- [13] Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.
- [14] Amit A. Nanavati, Siva Gurumurthy, Gautam Das, Dipanjan Chakraborty, Koustuv Dasgupta, Souagata Mukherjea, and Anupam Joshi. On the structural properties of massive telecom call graphs: findings and implications. In *Proceedings of the 2006 ACM International Conference on Information and Knowledge Management (CIKM)*, pages 435–444, 2006.
- [15] Mark Newman. Scientific collaboration networks: I. Network construction and fundamental results. *Physical Review*, 64, 2001.
- [16] Mark Newman. Scientific collaboration networks: II. Shortest paths, weighted networks, and centrality. *Physical Review*, 64, 2001.
- [17] Mark Newman. The structure and function of complex networks. *SIAM Review*, pages 167–256, 2003.
- [18] Andrea Ordanini and Kenneth L. Kraemer. Medion: the retail orchestrator in the computer industry. 2006.
- [19] Vinayaka Pandit, Natwar Modani, Sougata Mukherjea, Amit A. Nanavati, Sambuddha Roy, and Amit Agarwal. Extracting dense communities from telecom call graphs. In *Proceedings of the Third International Conference on Communication System softWare and MiddlewaRE (COMSWARE 2008)*, pages 82–89, 2008.
- [20] Bikram Sengupta, Satish Chandra, and Vibha Sinha. A research agenda for distributed software development. In *28th International Conference on Software Engineering (ICSE)*, pages 731–740, 2006.
- [21] Vibha Sinha, Bikram Sengupta, and Satish Chandra. Enabling collaboration in distributed requirements management. *IEEE Software*, 23(5):52–61, 2006.
- [22] Giuseppe Valetto, Mary Helander, Kate Ehrlich, Sunita Chulani, Mark N. Wegman, and Clay Williams. Using software repositories to investigate socio-technical congruence in development projects. In *Fourth International Workshop on Mining Software Repositories (MSR)*, page 25, 2007.
- [23] Stanley Wasserman and Katherine Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, New York, 1994.
- [24] T. H. Wei. *The algebraic foundations of ranking theory*. Cambridge University Press, London, 1952.