Optimal Resource Assignment in Workflows for Maximizing Cooperation

Akhil Kumar[1], Remco Dijkman[2], and Minseok Song[3]

[1] Smeal College of Business, Penn State University, University Park, PA 16802, USA
AkhilKumar@psu.edu
[2] Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
r.m.dijkman@tue.nl
[3] Ulsan National Institue of Science and Technology, UNIST-GIL 50, Ulsan 689-798, South Korea
msong@unist.ac.kr

Abstract. A workflow is a team process since many actors work on various tasks to complete an instance**.** Resource management in such workflows deals with assignment of tasks to workers or actors. In team formation, it is necessary to ensure that members of a team are compatible with each other. When a workflow instance of, say, an insurance claim (or a surgery) process is performed, the handoffs between successive tasks are often *soft* as opposed to *hard*, and actors who perform successive tasks in this process instance must cooperate. If they cooperate well, it can improve quality and increase throughput of the instance. In general, the degree of required cooperation between a pair of tasks varies and this should be captured by a model. This paper develops a model to capture the compatibility between actors while assigning tasks in a workflow to a group of actors. The model is tested through a simulation and the results from a greedy algorithm are compared with optimal results. A technique for computing the compatibility matrix is given and used for an empirical validation from a real execution log. We argue that workflow resource models should recognize soft handoffs and provide support for them.

1 Introduction

"We found that patients whose surgical teams exhibited less teamwork behaviors were at a higher risk for death or complications." [10]

Much work within organizations takes place in teams whether it is performing surgery (as in the quotation above), designing a car, or processing a customer's insurance claim application. Naturally, it is very important that members of a team, in addition to having the requisite qualifications, also be compatible with one another in order to ensure smooth execution and flow of the work. Of course, in a team of *n* workers or actors, it is not necessary that every pair of members must be fully compatible with each other, but the goal in general would be to maximize overall compatibility particularly across actors whose roles require considerable collaboration and cooperation. Non-cooperation can result in loss of productivity. In a similar vein, the need for optimization also arises in business processes. In a typical insurance claim process, several tasks must be done by different roles in a certain order. After a worker or actor completes her task she hands off the process workflow to the next actor. In a *hard handoff* no further interaction between the two actors may be required. But in a *soft handoff*, the two actors may still need to interact later for queries and clarifications even though the process definition may not reflect it. Thus, in practice "there is a series of *overlapping* and *nested* roles and responsibilities."[8] In general, an actor doing a later task in a workflow may need to refer back to consult with an actor who did a previous task for the same case. Hence, cooperation is necessary between the two actors of successive tasks so that the workflow can proceed smoothly.

Workflow management systems can be viewed from various perspectives such as: control flow, data flow and resource modeling. The control flow describes the ordering relationships between various tasks, and the data flow its data inputs and outputs. The resource model [7,19,18] refers to the roles and specific actors who are qualified to perform various tasks. Most resource assignment algorithms consider issues like suitability, urgency, conformance and availability [6, 11] while allocating tasks to actors. However, they fail to recognize the interactions among the actors performing different tasks in a workflow instance, say for insurance claim processing. In practice, there is need for such interaction.

The execution of a process instance, in general, is really a team effort involving multiple handoffs and the handoff should be as smooth as possible. The Free Dictionary (http://www.thefreedictionary.com) defines compatible as: "capable of existing or performing in harmonious, agreeable, or congenial combination with another." Thus, *compatibility* is a measure of the degree to which actors cooperate with one another in a workflow. Hence, compatibility between actors should be considered while assigning tasks to actors. Current approaches only consider suitability of an actor for a task in isolation of her compatibility with actors of others tasks in an instance. We propose a model that allows us to specify compatibilities among actors in compatibility matrix, and also the required degree of desired cooperation among tasks through a cooperation matrix. In general, compatibility between two actors may be task-specific, but for now we will assume that it is the same for all tasks.

As noted above, the medical domain is another area where multiple roles must work together in order to achieve a positive outcome, and compatibility and smooth coordination and handoffs between various personnel involved (such as surgeons, anesthesiologists, nurses, lab technicians, etc.) is very important [1,10]. In this paper we show how to model compatibility between actors while making work assignments so as to achieve a high degree of overall compatibility for the process. Section 2 gives a basic framework and preliminaries. Then, Section 3 describes our model for maximizing compatibility. Next, Section 4 gives a greedy heuristic and experimental results for its performance against an optimal solution. Sections 5 and 6 show how to compute the compatibility matrix and provide empirical validation respectively. Section 7 presents several directions for extending this approach. Finally section 8 gives a discussion along with related work and we conclude with Section 9.

2 Basic Framework

Consider an example of a medical insurance claim process model shown in BPMN in Figure 1. In this process, a claim is received, and then checked by a reviewer who verifies that it is a valid claim. Next, it is examined by an evaluator who determines the amount of the settlement. A manager must approve the claim, and finally the accounts officer issues a payment for it. Thus, the key steps or tasks, the roles that perform each step, the actors in the roles and their respective locations, in this process are:

Receive claim (role: customer service rep; actors: John, Mary; location: call center 1)
Review, validate, assign claim (role: reviewer; actors: Beth, Sue; location: call center 2)
Evaluate and determine settlement (role: evaluator; actors: Mike, Jim; location: client city)
Approve payment (role: manager; actors: Jen, Pat; location: regional office)
Make payment (role: accounts officer; actors: Mark, Lin; location: headquarters)

Notice that the roles for each task are geographically dispersed. This makes the need for cooperation even greater. As depicted by the dotted arrows in Figure 1, roles performing different tasks may need to interact. For example, after a claim is received, the reviewer might need to refer back to the customer service representative for clarification about certain missing information on the claim (say, the exact location or time of an accident is missing). Similarly, the evaluator may need to consult with the reviewer for additional details. Finally, the manager could seek clarifications with the evaluator regarding the payment amount before approving it.

Thus, even though formal representations for workflow processes may not show it, there is often a need for such referrals. But formal modeling approaches tend to neglect this issue. Our goal is to capture notions of compatibility between actors who will perform tasks where soft handoffs are important. Hence, a metric for compatibility is required.

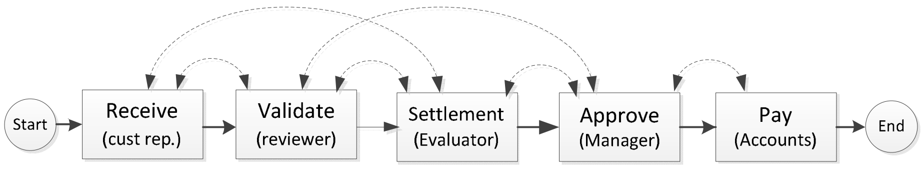

Fig. 1. A simplified insurance claim process with several tasks and roles (Dashed lines show the need for cooperation among actors of pairs of tasks)

Our metrics for compatibility within a team or a process workflow are:

$$Total\ Compatibility = \sum_{\forall(u1,u2,t1,t2)} fit_{u1,u2,t1,t2} * coop_{t1,t2} * cweight_{u1,u2}$$

$$Average\ Compatibility = \frac{Total\ Compatibility}{\sum_{t1,t2} coop_{t1,t2}}$$

Where $fit_{u1,u2,t1,t2}: \begin{cases} 1\ if\ actor\ u1, u2\ perform\ tasks\ t1, t2\ respectively \\ 0, otherwise \end{cases}$

$$coop_{t1,t2}: \begin{cases} 1, if\ cooperation\ required\ between\ tasks\ t1\ and\ t2 \\ 0, otherwise \end{cases}$$

$$cweight_{u1,u2}: compatibilty\ of\ actors\ u1, u2 \text{ on a continuous scale of 01}$$

The fit and cooperation values are stored in two matrices. Table 1 is an actor-actor compatibility matrix with values on a scale of 0 to 1 (from low to high compatibility). Table 2 gives a binary cooperation matrix for all pairs of tasks, where a '0' means cooperation between a pair of tasks is not required, and '1' that it is. We assume that cooperation is reciprocal; hence the matrices are symmetric. The start and end tasks are not shown in the tables because they are not performed by humans. The diagonal entries in the table are 1.0 to represent that an actor is fully compatible with herself. Here we assume that compatibility between actors is not task-specific, but to make it task specific, $cweight_{u1,u2}$ can be modified in the above formulation to $cweight_{u1,u2,t}$ where the new subscript *t* represent a task. Later we will show how this model is modified to allow non-discrete values of cooperation between actors.

Example 1: (Partial Cooperation) Below we calculate average compatibilities using the values in Tables 1 and 2. There are five main tasks in this instance. Moreover, as per Table 2 there are only 6 interactions where cooperation is required; hence it is a case of partial cooperation. Clearly, several combinations of actor assignments are possible here. Let us look at two examples.

Assignment 1 (random):
cust. rep: John ; Reviewer: Sue; Evaluator: Jim; Manager: Pat; accounts officer: Mark

Average compatibility = (0.1 + 0.3 + 0.6 + 0.6 + 0.1 + 0.1)/6 = 0.3

Assignment 2 (optimal):
cust rep: Mary; Reviewer: Beth; Evaluator: Jim; Manager: Jen; accounts officer: Mark

Average compatibility = (0.3 + 0.8 + 0.8 + 0.7 + 0.8 + 0.7)/6 = 0.683

The first assignment is made simply by randomly assigning a task to a qualified actor, while the second one is optimal. Clearly, there is a large difference (of more than 100%) in average compatibility between these two assignments.

Table 1. Actor-Actor Compatibility matrix (*cweight*)

Role > (Task) >	Cust. Rep (receive)		Reviewer (validate)		Evaluator (settle)		Manager (approve)		Accounts (pay)	
	John	Mary	Beth	Sue	Mike	Jim	Jen	Pat	Mark	Lin
John	1.0	0.9	0.8	0.1	0.8	0.3	0.9	0.3	0.4	0.2
Mary	0.9	1.0	0.3	0.7	0.2	0.8	0.9	0.2	0.1	0.8
Beth	0.8	0.3	1.0	0.8	0.3	0.8	0.7	0.3	0.2	0.9
Sue	0.1	0.7	0.8	1.0	0.9	0.6	0.4	0.6	0.8	0.4
Mike	0.8	0.2	0.3	0.9	1.0	0.9	0.3	0.9	0.8	0.1
Jim	0.3	0.8	0.8	0.6	0.9	1.0	0.8	0.1	0.3	0.9
Jen	0.9	0.9	0.7	0.4	0.3	0.8	1.0	0.8	0.7	0.3
Pat	0.3	0.2	0.3	0.6	0.6	0.9	0.8	1.0	0.1	0.8
Mark	0.4	0.1	0.2	0.8	0.8	0.3	0.7	0.1	1.0	0.9
Lin	0.2	0.8	0.9	0.4	0.1	0.9	0.3	0.8	0.9	1.0

Example 2:(Full Cooperation) Next consider a variation of the above example. Instead of assuming that cooperation between some pairs of participants is necessary, let us assume that all participants who work on an instance of a process must cooperate with each other. The corresponding cooperation matrix is shown in Table 3. The solutions from the random assignment and the optimal assignment are as follows.

Table 2. Cooperation matrix (*partial* cooperation required)

	Receive	Validate	Settle	Approve	Pay
Receive	–	1	1	0	0
Validate	1	–	1	1	0
Settle	1	1	–	1	0
Approve	0	1	1	–	1
Pay	0	0	0	1	–

Assignment 3 (random):
cust rep: Mary ; Reviewer: Sue; Evaluator: Jim; Manager: Jen; accounts officer: Lin
Average compatibility =
(0.7 + 0.8 + 0.9 + 0.8 + 0.6 + 0.4 + 0.4 + 0.8 + 0.9 + 0.3)/10= 0.66

Assignment 4 (optimal):
cust rep: Mary; Reviewer: Beth; Evaluator: Jim; Manager: Jen; accounts officer: Lin
Average compatibility =
(0.3 + 0.8 + 0.9 + 0.8 + 0.8 + 0.7 + 0.9 + 0.8 + 0.9 + 0.3)/10= 0.72

Table 3. Cooperation matrix (*full* cooperation required)

	Receive	Validate	Settle	Approve	Pay
Receive	–	1	1	1	1
Validate	1	–	1	1	1
Settle	1	1	–	1	1
Approve	1	1	1	–	1
Pay	1	1	1	1	–

In this example, the difference in average compatibility between the optimal and random assignments is much smaller than in Example 1. The improvement through an optimal reassignment of tasks in the full cooperation case is less because, in general, perhaps few actors cooperate well with all other actors in a process.

We have considered two scenarios involving different levels of cooperation. In general, the cooperation matrix could vary, and the best assignment will also be different accordingly. Next we describe our model for finding an optimal solution so as to maximize cooperation within the team.

3 Model – Optimal Work Assignment (OWA)

The objective of this model shown in Figure 2 is to maximize total (or average) *compatibility*. However, we express our objective function so as to minimize total *incompatibility* and the reason for this is explained shortly. Our notion of overall compatibility is as an aggregate of all pair-wise compatibilities between actors who are involved in task-pairs of a process that require cooperation. Since pair-wise actor-actor compatibility ranges between 0 and 1, *incompatibility=(1− compatibility).*

Model OWA

$$Minimize \sum_{u1,u2,t1,t2} fit_{u1,u2,t1,t2} * (1 - cweight_{u1,u2})$$

Subject to:

$\sum_u does_{u,t} = 1,\ \forall t$ (1)

$does_{u,t} \leq cando_{u,t}$ (2)

$does_{u1,t1} + does_{u2,t2} - fit_{u1,u2,t1,t2} \leq 1, \forall\, t1, t2\ where\ coop(t1,t2) = 1$ (3)

Where:

$$fit_{u1,u2,t1,t2} = \begin{cases} 1\ if\ actors\ u1, u2\ perform\ tasks\ t1, t2\ respectively \\ 0, otherwise \end{cases}$$

$$does_{u,t} = \begin{cases} 1, if\ actor\ u\ is\ assigned\ to\ perform\ task\ t \\ 0, otherwise \end{cases}$$

$$cando_{u,t} = \begin{cases} 1, if\ actor\ u\ is\ qualified\ to\ performtask\ t \\ 0, otherwise \end{cases}$$

$$coop_{t1,t2} = \begin{cases} 1, if\ cooperation\ is\ needed\ between\ t1\ and\ t2 \\ 0, otherwise \end{cases}$$

$cweight_{u1,u2}: compatibilty\ between\ actors\ u1, u2$

Fig. 2. Model OWA for optimal work assignment

By constraint 1 of the OWA model, every task must be assigned to exactly one actor. The second constraint requires that the actor *u* who is assigned to perform task *t*($does_{u,t}$) must be qualified to do it ($cando_{u,t}$). The third constraint forces the fit variable between two actors doing tasks that have a soft handoff between them to 1. Thus, the fit variable $fit_{u1,u2,t1,t2}$ must be 1. The model in Figure 2 is called an integer programming (IP) formulation which is known to be NP-complete [2]. Hence solving the OWA is an NP-complete problem too. It can be solved with a tool like CPLEX [4]. The solution of the model gives the optimal assignment by finding the values for the variable $does_{u,t}$ for all *u,t*. Additional constraints can be added to this basic model to enforce minimum (maximum) limits on number of tasks assigned to any actor. Note that the objective function assumes that overall compatibility is linear in individual actor-pair compatibility.

Now, the reason the objective function minimizes total incompatibility is as follows: If we try to maximize compatibility then the *fit variables are all forced to 1* resulting in an incorrect formulation. However, if we express the objective as minimizing incompatibility this error does not occur, and a fit variable $fit_{u1,u2,t1,t2}$ assumes a 1 value only when actor *u1* does task *t1* and actor *u2* does task *t2*.

To construct the OWA model, we only need to know the data in the coop, cweight and cando matrices. The complexity of this problem is $O(t^u)$, where *t* is number of tasks and *u* is number of actors per task. In the next section we develop a heuristic to solve this problem.

4 A Greedy Heuristic and Results

Here we describe a greedy heuristic to solve the actor assignment problem. The main steps are shown in Figure 3. The coop, cweight and cando arrays are taken directly from the model described above. For each successive task *t1* (line 1), we consider each actor *u1* (line 2) who can do *t1*. Then, for each actor *u1* and for every other task *t2* (line 3) such that cooperation between *t1* and *t2* is required (line 4), we find the maximum compatibility actor with actor *u1* from the cweight array and accumulate the compatibility in a variable *score*. This is repeated for every task *t2* that requires cooperation with *t1* and in this way a score is computed for each actor who can perform task *t1*. Finally, the actor with the maximum score is assigned the task *t1* (line 7) and corresponding actor to other tasks that need cooperation with t1. This is repeated until all actors are assigned. The algorithm returns the assign array.

```
Algorithm Greedy_Coop
       Input: coop[][],cweight[][], cando[][]
       Output: assign[]
1     for each (task t1 = 1,…, num_tasks)
2       for each (u1 ∈ cando[t1])
3         for each (task t2 = t1+1,…, num_tasks)
4     if (coop(t1,t2)&& not(assign[t1])&& not(assign[t2]):
        score(u1)=score(u1) + max(cweight(u1,u2),
                                    ∀u2∈cando[t2])
5         end for
6       end for
7       assign[t1] = u*,
            s.t. score[u*],u*= max(score[u], u∈cando[t1])
8       for each (task t2 = t1+1,…,num_tasks)
9      if (coop(t1,t2)&& not(assign[t2]):
        assign[t2] = u2*, s.t.
         cweight[u*,u2*]=max(cweight[u*,u2], u2∈cando[t2])
10     end for
11     return(assign[])
```

Fig. 3. A greedy heuristic for actor assignment

This is a greedy algorithm. At each successive step, we assign actors to a task based on the best compatibility for this particular assignment without optimizing across all tasks. Next, we compare the greedy heuristic results against the optimal solution to see how much improvement is possible by using the optimal approach.

We conducted experiments to compare the greedy heuristic with the optimal solution. The greedy heuristic was implemented in Python language, while the optimal solution was found by solving the model using CPLEX software [4]. We used a simulation first to create the data for the experiments, and the parameters of the simulation are given in Table 4. In particular there are 10 tasks in the process and 20 actors. Each task can be done by either 2 or 3 actors. First, we select the number of qualified actors for each task (2 or 3, with equal probability), and then pick the actual actors at random. Our cooperation model assumes that:

(1) task *i* must cooperate with the next task *i+1* with probability 1.0

(2) tasks in the pairs (*i,i+1*) and (*i,i+2*) must cooperate with probability 0.5

Finally an actor-actor compatibility matrix is generated where compatibility values of 0.1, 0.2, …, 0.9 are randomly assigned. If the same actor performs two tasks, then her compatibility with herself is 0.99 (i.e. close to 1). In these experiments we assumed availability of all actors was 1, i.e. they were all available.

Table 4. Parameters used in the simulation experiment

Parameter	**Description**	**value**
# tasks	Number of tasks	10, 20
Total # actors	Number of actors	20, 40
Task- actor assignment	For each task, assign actors who can perform the task	Pick 2 or 3 actors at random
Cooperation requirement	Between tasks *i* and *i+1* with prob. 1, and between *i,* and, *i+2* and *i+3*, with prob. 0.5	
Compatibility weight	Weight between 0 and 1 to measure degree of fit between two actors where handoff is important	0.1,0.2, … 0.9
Availability	Extent of availability of an actor (0.0,…1.0)	1.0

In Table 5 (a) we summarize the results for 10 cases with 10 tasks and 20 actors in each case. The actual actor assignments produced by the heuristic are not shown. In case 1, the heuristic produces an assignment where task 1 is assigned to actor 19, task 2 to actor 10, and so on. For this case the heuristic assignment is very surprisingly close to the optimal solution. In fact 9 out of 10 actor assignments are the same except that task 9 is assigned to actor 10 in the heuristic instead of actor 7. We also report the average compatibility, i.e. the average of the compatibility values across the '1' entries in the cooperation matrix, along with the percentage gap between the optimal and the heuristic solutions. In case 1, the heuristic is worse than the optimal by just about 6%, but in other cases, the gap is larger, even as high as 40% in case 9. Overall, across all 10 cases the average gap is about 19%.

Similarly, the results for a second experiment with 20 tasks and 40 actors are given in Table 5 (b). Now there is an average gap of 17% between the performance of the optimal and the heuristic, and it lies between 8% (case 3) and 23% (case 7). In case 3, 6 out of 10 actor assignments are the same, while in case 7, 5 out of 10 are the same.

The results clearly show that the greedy algorithm is useful but suboptimal. The main problem observed in both sets of experiments with the greedy algorithm is that if it makes a bad assignment early on, this effect gets magnified with successive task assignments. Thus, it can lead to a very inferior final solution since there is no backtracking in the greedy algorithm. The assignment of actors to tasks can be done dynamically rather than making a static assignment at the start of the process instance. Thus, in a dynamic mode an initial assignment is made at the start, and after each successive task is completed, the algorithm is rerun to make the next assignment based on availability of actors.

The experiments were carried out on a typical desktop PC (Intel dual core CPU at 2.40 GHz with 3.25 GB RAM) running CPLEX. The running times to find the solutions were in fractions of a second for the problems above. Thus, for problems of

Table 5. Results for average compatibility: heuristic vs. optimal solutions

(a) 10 tasks, 20 actors

	Avg. Compat.		% gap
Case	Greedy	Opt.	
1.	0.656	0.700	6.29
2	0.650	0.759	14.36
3.	0.669	0.760	11.97
4.	0.653	0.785	16.82
5.	0.591	0.740	20.14
6.	0.615	0.737	16.55
7.	0.461	0.597	22.78
8.	0.550	0.761	27.73
9.	0.466	0.780	40.26
10.	0.615	0.730	15.75
Avg.	0.593	0.735	19.32

(b) 20 tasks, 40 actors

	Avg. Compat.		% gap
Case	Greedy	Opt.	
1.	0.615	0.684	10.09
2	0.568	0.717	20.78
3.	0.557	0.607	8.24
4.	0.605	0.759	20.29
5.	0.608	0.712	14.61
6.	0.596	0.771	22.70
7.	0.567	0.734	22.75
8.	0.556	0.718	22.56
9.	0.570	0.691	17.51
10.	0.619	0.720	14.03
Avg.	0.586	0.711	17.36

medium size one can find optimal solutions but for larger problems heuristic methods may be more appropriate.

5 Automatically Computing the Compatibility Matrix

To fully benefit from optimal work assignment with the compatibility matrix, a compatibility matrix must be determined that corresponds to the manner in which actors work together in practice. While this matrix can be designed in a traditional manner, e.g. based on interviews with the actors, this is not ideal. In particular, because it is unlikely that the interviewees will accurately report on their cooperation with others, due to political considerations. Therefore, we propose an approach in which we derive the compatibility matrix automatically based on an execution log that contains for each execution trace: the executed tasks, the actor executing each task and the total throughput time.

The basic idea is that if, in cases where two actors u_1 and u_2 cooperate, the throughput time is lower on average than in the general case, these actors can be assumed to have a higher compatibility. Conversely, if the throughput time is higher on average, the actors can be assumed to have a lower compatibility. Based on this assumption, we can use a sigmoid function to derive the actors' compatibility from the throughput times as follows. Given two actors u_1 and u_2, the average throughput time t of the process, and the average throughput time tc of the process for execution traces in which u_1 and u_2 collaborated,

$$cweight_{u1,u2} = \frac{1}{1 + e^{-k(t-tc)}}$$

Figure 4 illustrates the relation between t, tc and *cweight* for $k = 1$. In this function k is a parameter that we can vary to obtain better results. In particular, if the variance in throughput time is high k should be smaller to be more sensitive to these variances, similarly, if the variance is low, *k* should be greater. A suggestion is, to choose *k* such that the sigmoid is most sensitive for *tc* from the first to the third quartile of the throughput times domain (see Figure 4). Given the first quartile is *q*1, the third quartile *q*3 and the average *t*, *k* = 10/(*q*3-*q*1).

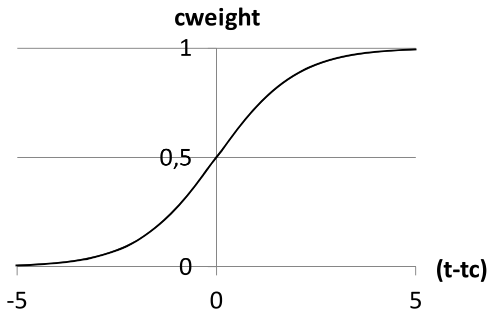

Fig. 4. Relation between *(t– tc)* and *cweight* in a collaboration

Alternatives to the sigmoid function, such as a simple linear function, and alternative values of k can also be used to compute the compatibility matrix. We experimented with some different values for k during the evaluation (see Section 6), but did not evaluate alternatives exhaustively. In future work, we aim to investigate alternative functions and determine the parameter settings (a value for *k* in case of the sigmoid function) and a function that produces the best result.

Figure 5 shows an example of the automated computation of the compatibility matrix from an execution log. The average throughput time of the execution traces is 9 and the average throughput times for traces where a particular combination of actors appears is shown in Figure 5 (b). For example, the average throughput time for execution traces in which John and Mary work together is 8.5, for traces where John and Beth work together it is 10. Figure 5 (c) shows a compatibility matrix computed based on the throughput times using the sigmoid function. For example, the *cweight*

(a) execution traces

Trace	**Receive**	**Validate**	**Settlement**	**t**
1	John	Mary	Mike	8
2	John	Beth	Mike	10
3	John	Mary	Mike	9

(b) average throughput times

	John	Mary	Beth	Mike
John	9	8.5	10	9
Mary	8.5	9	–	8.5
Beth	10	–	9	10
Mike	9	8.5	10	9

(c) compatibility matrix

	John	Mary	Beth	Mike
John	0.5	0.6	0.3	0.5
Mary	0.6	0.5	–	0.6
Beth	0.3	–	0.5	0.3
Mike	0.5	0.6	0.3	0.5

Fig. 5. Example of automated computation of the compatibility matrix

for the collaboration between John and Mary is $1/(1 + e^{-(9-8.5)}) \approx 0.6$, i.e.: John and Mary are slightly more compatible than the average, which is 0.5, and certainly more compatible than John and Beth, who have compatibility $1/(1 + e^{-(9-10)}) \approx 0.3$.

In the next section, we will apply the automatic computation of the compatibility matrix, as it is explained here, to an execution log from practice.

6 Empirical Evaluation

We evaluated the technique described in this paper using an execution log of a doctor's consultation process in Seoul National University Bundang Hospital, South Korea. The log was manually constructed from data that was extracted from the software systems that are used in the various process steps. The process involved five steps: reserving a room for the consultation; the actual consultation; planning follow-up appointments; making payment; and issuing a prescription. The first and second steps are performed by the same role (the doctor), which has 174 possible actors. The third step is performed by a secretary, which has 74 possible actors. The fourth and fifth steps are again performed by the same role (an administrator) and had 38 possible actors. We had 4,446 execution traces.

First, we empirically validated that collaborations between actors did indeed have an effect on the throughput time. Because of the large number of unique collaborations, we focused on a subset of collaborations that occurred more than 20 times, and disregarded other collaborations as insignificant (in fact, many occurred only once). We also focused on collaborations in the third, fourth and fifth steps of the process. These steps involved administrative tasks around the consultation: making the next appointment, receiving a prescription and paying for the appointment. It was felt that these steps were more likely to be affected by compatibility and less likely to be affected by other factors, such as complexity of the medical case. This selection resulted in 35 pairs of collaborations, associated with 1,717 execution traces. The data was analyzed in SPSS. We determined whether the throughput times for the collaborations were normally distributed, using a Shapiro-Wilk test. The test showed that the data was *not* normally distributed. Consequently, we used a Kruskal-Wallis test (instead of ANOVA) to determine whether the collaborations differed significantly, which was found to be the case at a 0.05 significance level. Therefore, we conclude that there are significant differences in throughput times between collaborations.

Second, we evaluated the theoretical improvement that the technique described in this paper can achieve in work assignments. We did so by determining the compatibility matrix for the case and subsequently determining the optimal work assignment for this compatibility matrix. We used the sigmoid function to determine *cweight* with the parameter *k* set such that the function was most sensitive in the second and third quartiles of the throughput times. Figure 6 shows a part of the compatibility matrix for the case, showing the actors in the process and their compatibility. The actors are represented by codes such as EIC, CDCJJ, etc. to ensure anonymity. Due to the large number of actors involved in the case, the full compatibility matrix has 286 x 286 cells. The optimal work assignment computed from the compatibility matrix leads to an average throughput time of 6 minutes, which is a strong improvement over the overall average throughput time of 42.9 minutes. However, this average is based on *cweights* computed from only one execution trace for illustration of an extreme case.

Focusing on assignments that were based on at least 10 execution traces, the best work assignment leads to an average throughput time of 23.7 minutes, still a strong improvement over the overall average throughput times. Interestingly, in this case the third-best work assignment is actually better at an average of 19.7 minutes. The best assignment based on at least 20 execution traces has an average throughput time of 26.9 minutes.

	EIC	CDCJJ	CHBAB	CEFGG	…
EIC	0.50	0.99	–	–	…
CDCJJ	0.99	0.50	0.25	0.75	…
CHBAB	–	0.25	0.50	–	…
CEFGG	–	0.75	–	0.50	…
…	…	…	…	…	…

Fig. 6. Part of the compatibility matrix of the case

Although our results on throughput were not tested for statistical significance, we believe these differences are too large to be explained by differences in worker competence alone, especially given that the tasks involved are of low complexity. Hence, our initial evidence points to varying levels of cooperation among actors.

7 Further Extensions

In this section we consider some variants of the basic model. The first one allows us to model varying degrees of cooperation between actors instead of just 0-1 binary cooperation. The second extension considers how to find an optimal assignment when multiple paths exist in the process. Finally, the last variant includes cost in the model as a constraint or an objective.

7.1 Varying Degrees of Required Cooperation

In the discussion thus far, the coop matrix only contained discrete 0-1 entries for pairs of tasks, where a 0 indicated cooperation was not required between the actors performing two tasks, and a 1 indicated it was required. In general, varying degrees of cooperation may be required between actors of different pairs of tasks. For example, in the process of Figure 2, a high degree of cooperation (say, 0.9) may be necessary between the evaluator and the manager, the need for cooperation between the manager and the accounts officer may be less (say, 0.3). This can be captured by associating a continuous parameter between 0 and 1 to denote the strength of cooperation required between the performers of two tasks. Thus, the cooperation matrix would contain $coop_{t1,t2}$ entries that are values between 0 and 1, and not binary values. These values would be determined subjectively by somebody with knowledge about the process. Again, the objective function would also be modified as follows:

$$Minimize \sum_{\forall u1,u2,t1,t2} coop_{t1,t2} fit_{u1,u2,t1,t2} * (1 - cweight_{u1,u2})$$

The rest of the formulation would remain unchanged.

7.2 Multiple Paths in a Process

The process described in Figure 1 is linear. Now, consider a modified version of that process as shown in Figure 4 with two alternative branches after the *validate* step, the lower branch being taken when the claim is rejected outright on initial review, say if it is not covered by the policy. To handle this situation, we modify the objective function by introducing a new parameter $p_{t1,t2}$ for the transition probability between two tasks t1, t2. An example of a transition probability matrix is shown in Table 6. The $p_{t1,t2}$ values are also shown on the edges in Figure 7. The revised objective function is a weighted sum of the probability of path $\pi_i$being taken and the incompatibility along that path. The probability $P(\pi_i)$of a path being taken is computed as the product of probabilities along all the edges on the path from the start node to t2. The constraints, however, remain the same as before. Thus, the new objective function is:

$$Minimize \sum_{\pi_i \in Paths} P(\pi_i) \sum_{u1,u2,t1,t2} fit_{u1,u2,t1,t2} * (1 - cweight_{u1,u2})$$

In a process model with loops the compatibility matrix is created as before by considering the activities in a loop and the compatibility requirements for them. An estimate is used for the average number of loop repetitions, so the pairs of activities within the loop can be weighted by this factor in the objective function.

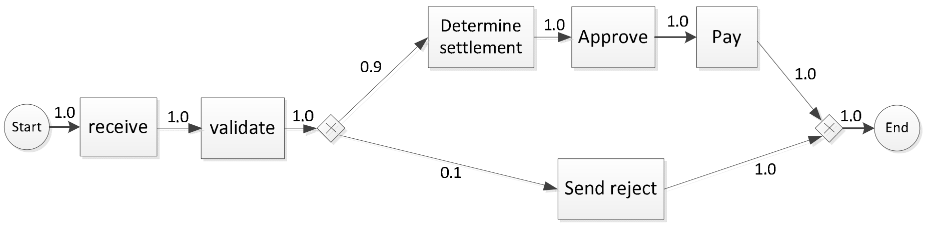

Fig. 7. A revised process (labels on arcs show transition probabilities)

Table 6. A task-task transition probability matrix for the process in Figure 4

	Receive	Validate	Settle	Approve	Pay	Reject
Receive	0	1	0.9	0.9	0.9	0.1
Validate	0	0	0.9	0.9	0.9	0.1
Settle	0	0	0	1	1	0
Approve	0	0	0	0	1	0
Pay	0	0	0	0	0	0

7.3 Optimization of Cost, Time, Resource

Model OWA in Section 3 has been formulated to minimize incompatibility, or equivalently, to maximize compatibility. However, in some actor assignment scenarios an additional objective is to minimize cost, time or another resource. Given an actor-task cost matrix, where $cost_{u,t}$ is the cost of actor u performing one instance of task t, the model can be modified easily. In such a case it is possible to incorporate cost into our model as an additional constraint such as:

$$\sum_{u,t} cost_{u,t} does_{u,t} < cost_{max}$$

Where, $cost_{max}$ is the maximum allowable cost for the assignment.

Another alternative is to convert the objective function and add it as a constraint into a cost optimization model. Then the constraint is expressed as:

$$\sum fit_{u1,u2,t1,t2} * (1 - cweight_{u1,u2}) > 1 - compat_{min}$$

where $compat_{min}$ is a minimum desired compatibility threshold.

In a similar way, it is possible to further extend the formulation by adding availability, throughput and resource consumption constraints.

8 Discussion and Related Work

Research has shown that cooperative behavior is reciprocal [14] as we assume here, and it affects service quality and performance [17]. Therefore, a workflow framework should incorporate cooperation by providing suitable constructs for modeling compatibility between actors. This means that a process designer should be able to model soft handoffs between tasks and the degree of cooperation. This information can be used in making resource assignments to enable a smoother flow of work.

In general, the time to perform a task consists of two elements: the intrinsic capability of an actor(u_i) to perform task t based on skill and experience, and her compatibility(u_i,u_j) with other actors (u_j) of related tasks of an instance. Thus,

Avg. time for task t by actor u_i = f(capability(u_i,t), compatibility(u_i, u_j))

Although we do not model the output quality of a process instance in this paper, it is reasonable to assume that given qualified actors, better compatibility among them will lead to higher quality and greater customer satisfaction. Further work is needed to develop a more elaborate model that can capture quality and tease out the role of the capability and compatibility elements. The approach described here can be implemented with a push-pull hybrid strategy. Actors would be offered a list of new tasks based on their compatibility, and then they may accept tasks from it.

In recent years there has been a surge of interest in modeling, connecting, scheduling and optimizing business processes both within and across organizations [16,18,19]. All such processes involve actors interacting in a collaborative manner. Techniques for organizational mining to discover organizational models and social networks are discussed in [15]. These models can assist in improving the underlying processes and provide insights for resource assignment. When many actors or workers collaborate on a team or on an instance of a running workflow, several factors can influence the overall performance. In [12], based on an extensive empirical study it was shown that there is a positive effect on performance of workflow instances when actors are located geographically close together. This study has implications for assignment of work to distributed actors, and in relation to our work it suggests that geographical distribution of actors may affect cooperation adversely.

The issue of cooperation among actors also has implications for best practices in business process redesign [9]. In a cooperative setting, a process may be designed in such a way that the boundaries between tasks are flexible. In a non-cooperative environment the interfaces between tasks must be rigid. In [13] it is shown that asymmetry in task size of tasks in a process, knowledge intensity levels and required customization needs of tasks have an impact on throughput times and are factors to consider in process redesign. When knowledge intensity and level of customization are high, effective communication becomes critical in ensuring a smooth handoff, and hence compatibility between the actors carrying out the handoff is important. It can also be helpful to develop handoff protocols for better performance of a process as was shown for the case of nursing shift handoffs in critical care [1].

There is related work as well on assignment of tasks to actors. Wolf [18] describes a constraint programming approach for modeling and scheduling clinical pathways. An IP formulation with the objective to minimize cost for assigning medical personnel is discussed in [3]. Another approach for assigning work in emergency situations [11] is based on *threshold models* consisting of two components, threshold and stimulus. As stimulus associated with a task increases, even actors who have a high threshold for performing the task respond. Since cooperation plays an important role in emergencies, compatibility should be a factor in deciding the stimulus.

9 Conclusions

In this paper we have highlighted the importance of compatibility among actors for resource assignments in workflows. In practice, the actors who participate in a workflow instance are part of a collaborative team. Empirical evidence from an execution log of a doctor's consultation process in a hospital was given to show that throughput times can vary considerably when resource assignments change. Thus, there is a need to adequately model soft handoffs between tasks. Such situations are frequent in practice and this issue has received little attention in research literature. We developed a novel approach for such scenarios using the notion of compatibility between tasks, and built a formal model to describe assignments of actors to tasks so as to maximize overall compatibility across an end-to-end workflow instance. The optimal solution for this model performed 20% better than a heuristic greedy algorithm. For medium size problems the optimal solution could be found very fast. A technique for discovering compatibility matrices from logs was described, but it needs further validation. Other non-greedy heuristics for task assignment would also be worth exploring.

We argue for new constructs for modeling of soft handoffs that allow cooperation among actors and sharing of responsibility across tasks in a workflow. Future work should examine ways to model such cooperation more accurately, and also study its impact on throughput and other metrics of performance. More research is also needed to understand and better model factors that affect cooperation.

Acknowledgement. Song was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2011-0010561).

References

1. Berkenstadt, H., Haviv, Y., Tuval, A., et al.: Improving handoff communications in critical care: Utilizing simulation-based training toward process improvement in managing patient risk. CHEST 134(1), 158–162 (2008)
2. Garey, M.R., Johnson, D.S. (eds.): Computers and Intractability: AGuide to the Theory of NP-Completeness. W.H. Freeman, San Francisco (1979)
3. Grunow, M., Günther, H.-O., Yang, G.: Development of a decision support model for scheduling clinical studies and assigning medical personnel. Health Care Management Science 7(4), 305–317 (2004)
4. ILOG: Ilog CPLEX software, Version 11.010 (2008)
5. Jablonski, S., Bussler, C.: Workflow Management: Modeling Concepts, Architecture and Implementation. Thomson Computer Press, London (1996)
6. Kumar, A., van der Aalst, W.M.P., Verbeek, H.M.W.: Dynamic work distribution in workflow management systems: How to balance quality and performance. Journal of Management Information Systems 18(3), 157–193 (2002)
7. Kumar, A., Wang, J.: A framework for designing resource driven workflow systems. In: Rosemann, M., vom Brocke, J. (eds.) The International Handbook on Business Process Management, pp. 419–440. Springer (2010)
8. Leach, L., Myrtle, R., Weaver, F., Dasu, S.: Assessing the performance of surgical teams. Health Care Manage Rev. 34(1), 29–41 (2009)
9. Mansar, S., Reijers, H.: Best practices in business process redesign: validation of a redesign framework. Computers in Industry 56(5), 457–471 (2005)
10. Mazzocco, K., Petitti, D.B., Fong, K.T., Bonacum, D., Brookey, J., Graham, S., Lasky, R., Sexton, J., Thomas, E.: Surgical team behaviors and patient outcomes. The American Journal of Surgery 197(5), 678–685 (2009)
11. Reijers, H.A., Jansen-Vullers, M.H., Zur Muehlen, M., Appl, W.: Workflow management systems + swarm intelligence = dynamic task assignment for emergency management applications. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 125–140. Springer, Heidelberg (2007)
12. Reijers, H.A., Song, M., Jeong, B.: Analysis of a collaborative workflow process with distributed actors. Information System Frontiers 11(3), 307–322 (2008)
13. Seidmann, A., Sundararajan, A.: The effects of asymmetry on business process redesign. International Journal of Production Economics 50, 117–128 (1997)
14. Sen, S.: Reciprocity: a foundational principle for promoting cooperative behavior among self-interested actors. In: Proceedings of the Second International Conference on Multiactor Systems, pp. 322–329. AAAI Press, Menlo Park (1996)
15. Song, M., van der Aalst, W.M.P.: Towards comprehensive support for orga-nizational mining. Decision Support Systems 46(1), 300–317 (2008)
16. Sun, S., Kumar, A., Yen, J.: Merging workflows: A new perspective on connecting business processes. Decision Support Systems 42(2), 844–858 (2006)
17. Tjosvold, D., Moy, J., Sasaki, S.: Co-operative teamwork for service quality in East Asia. Managing Service Quality 9(3), 209–216 (1999)
18. Wolf, A.: Constraint-based modeling and scheduling of clinical pathways. In: Larrosa, J., O'Sullivan, B. (eds.) CSCLP 2009. LNCS, vol. 6384, pp. 122–138. Springer, Heidelberg (2011)
19. ZurMühlen, M.: Organizational management in workflow applications – Issues and perspectives. Information Technology and Management 5(3-4), 271–291 (2004)