INFORMS Journal on Computing
Vol. 17, No. 3, Summer 2005, pp. 374–387
ISSN 1091-9856 | EISSN 1526-5528 | 05 | 1703 | 0374

DOI 10.1287/ijoc.1040.0067

Effective Role Resolution in Workflow Management

Daniel D. Zeng, J. Leon Zhao
Department of Management Information Systems, Eller College of Management, University of Arizona, Tucson, Arizona 85721, USA {zeng@eller.arizona.edu, lzhao@eller.arizona.edu}

Workflow systems provide the key technology to enable business-process automation. One important function of workflow management is role resolution, i.e., the mechanism of assigning tasks to individual workers at runtime according to the role qualification defined in the workflow model. Role-resolution decisions directly affect the productivity of workers in an organization, and consequently affect corporate profitability. Therefore it is important to develop effective policies governing these decisions. However, there has not been a formal treatment of role-resolution policies in the literature. In this paper, we analyze role-resolution policies used in current workflow practice and propose new optimization-based policies that utilize online batching. Through a computational study, we examine three workflow-performance measures including maximum flow-time, average workload, and workload variation under these policies in different business scenarios. These scenarios vary by overall system load, task-processing-time distribution, and the number of workers. Based on computational results, we obtain the following insights that can help guide the selection of role-resolution policies. (a) As the overall system load increases, the benefit of using batching-based online optimization policies becomes more significant. (b) Processing-time variation has a major impact on workflow performance, and higher variation favors optimization-based policies. (c) Online optimization has the potential to reduce average workload significantly, and to reduce workload variation significantly as well.

Key words: workflow management; role resolution; online batching
History: Accepted by Amit Basu, Area Editor for Knowledge and Data Management; received January 2003; revised June 2003; accepted November 2003.

1. Introduction

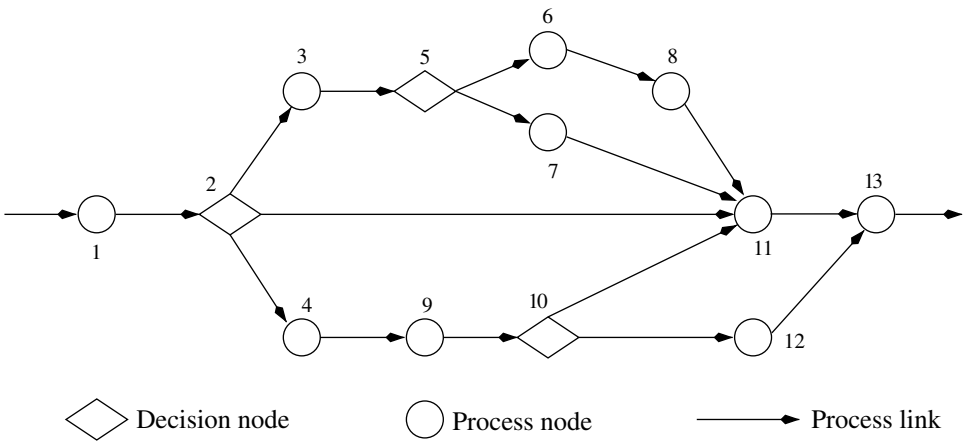
1.1. Role Resolution in Workflow Management

Internet technology has brought about many new opportunities to conduct business electronically and has made process automation a critical initiative of modern corporations, leading to e-business (Pinker et al. 2002). One important trend in e-business is to use workflow-management systems to automate business processes in both modeling and execution (Aissi et al. 2002). As a result, the runtime efficiency of workflow systems can have a far-reaching effect on enterprise competitiveness, given the breadth and depth of business-process automation in all modern corporations.

A critical function of a workflow system is to allocate tasks to workers that have been assigned to perform the given types of tasks (Stohr and Zhao 2001). To illustrate the concepts of job design and task allocation, we discuss a workflow model for claim handling, which was extracted from a business case involving a large manufacturer (Zhao and Stohr 1999). This claim-handling workflow, illustrated in Figure 1, captures the activities as part of the customer service function of the manufacturer. When a customer makes a claim about a potentially defective product, the reception desk records the claim. The claim is then classified into one of three work types: repair, refund, and field work. Depending on the work type, the claim is handled by different work groups through different routes.

This claim-handling workflow includes 3 decision nodes and 10 process nodes. In our definition, a process node is a point in the workflow where tasks are performed. A decision node is a special process node that leads to two or more branches. The specific activity associated with each node is specified in Figure 1. Although not explicitly shown, each node (either process or decision) contains a work list (or task queue) and a number of qualified workers who can perform the related tasks.

For instance, Node 4 corresponds to the task of "field inspection." This task requires workers with a broad knowledge of company products and a lot of experience to diagnose potential problems quickly. Similarly, other nodes in the workflow also require a unique set of skills and knowledge based on the nature of the associated tasks. Consequently, the manufacturer needs to staff an adequate number of workers with the right qualifications for each task. At the operational level, the workflow system is expected to



1 Reception desk
2 Work classification (repair, refund, or field work)
3 Inspection and diagnosis before repair
4 Field inspection
5 Deciding how to repair
6 Factory repair
7 Workshop repair
8 Factory inspection after repair
9 Field testing
10 Decision on payment
11 Accounting
12 Inspection by a manager
13 Final inspection and delivery

Figure 1 A Workflow Model for Claim Handling

automate the process of matching tasks with workers to achieve high productivity.

In general, based on their organizational goals and business functions, organizations make hiring and training decisions periodically to ensure that an adequate number of workers with the right combination of skills are available (van den Beukel and Molleman 2002). From the perspective of workflow management, these decisions translate into a functional role design, which assigns one or more specific roles to each worker. We call the decisions made in this step the *staffing* decisions.

In the runtime phase of workflow execution, tasks are assigned to workers who are qualified to perform the roles determined by the staffing decisions. In the workflow literature, the act of assigning a task at runtime to an appropriate worker is referred to as *role resolution* (Cheng 2000). For instance, the task associated with Node 3 in the above claim-handling example is "inspection and diagnosis before repair" and must be performed by workers who are assigned the "quality-assurance" role. However, because of variations in training, experiences, and other capabilities, each worker may take a different amount of time for a given task. In addition, some workers might be already occupied with other tasks, thus affecting the completion time of the task. As a result, different ways of assigning the tasks at runtime can lead to significantly different efficiencies for the quality-assurance group. That is, there is an opportunity to achieve more effective operation by using an appropriate role-resolution policy in a workflow system. This is especially important in a scenario where a large volume of tasks arrive in realtime and need to be processed within a given time period.

In sum, staffing decisions are typically made offline periodically and are of a *strategic* nature, while role-resolution decisions need to be made online and represent a major *operational*-level decision. Theoretically, it is possible to develop a *global* optimization model that captures and makes all related staffing and role-resolution decisions in an integrated manner. Practically, however, applying this integrated model poses significant challenges. First, in most workflow practice, staffing decisions and role-resolution decisions are being made in isolation and according to different time scales. Changing this paradigm means drastic redesign of the business process. Second, the integrated model will be computationally intractable. A related consideration is that this model may need to be invoked in realtime. An overblown model that takes a long time to compute and may entail reassigning people to different roles on the fly can be problematic.

These considerations motivate us to focus this paper on the operational-level role-resolution decisions. We assume that staffing decisions have already been made. Taking as input these staffing decisions, we model and compare various role-resolution policies applicable to a wide range of workflow applications. We aim to uncover managerial implications of various role-resolution schemes on overall

workflow effectiveness and efficiency. The ultimate goal of our research is twofold: (a) to develop effective role-resolution policies that can be applied in various business scenarios, and (b) to provide guidance to the design and development of advanced role-resolution mechanisms in next-generation workflow technology. In §§1.2 and 1.3, we first present general research issues in workflow role resolution from a modeling perspective, and then discuss the state of the art of role resolution in commercial workflow systems.

1.2. Modeling Role Resolution

Broadly speaking, any role-resolution model consists of two main components, *task modeling* and *worker modeling*. Task modeling is mainly concerned with the following issues.

Task Arrivals. In certain applications, task information is completely known at the beginning of workflow execution. In other applications, however, this is not the case. For instance, in the claim-handling example discussed in §1.1, claims are arriving dynamically in a random manner. For the incomplete-information case, how much information about the tasks is available (e.g., whether the probability distribution of task arrivals is known) has major impact on role-resolution decision making.

Task Dependences. There exist many types of dependences among tasks, ranging from no dependences, simple precedence constraints, to the full-fledged project-network type of precedence constraints among tasks, which specifies which tasks can be run in parallel, and which tasks must be sequentialized (Morton and Pentico 1993). These dependences are usually modeled by the workflow definition and taken as input by the role-resolution module that considers overall workflow effectiveness.

Resource Needs. In some applications, a complex task can only be processed by a team of workers collaboratively. Each team member is required to possess a particular set of skills. In other applications, each task needs only one qualified worker. In many applications, accomplishing some tasks may also require the worker to use common resources (e.g., specialized machinery or a limited-seat software license) shared among workers.

Deadlines. In some workflow applications, tasks have explicit deadlines. Missing a deadline carries various types of economic penalties (e.g., late fees charged proportionally to the extent of lateness). In other systems, tasks do not have deadlines, although quicker turnaround is always preferred. In some applications, both types of tasks may be present.

There are two main issues related to worker modeling.

Capacity. In some cases, a worker can only work on one task at a time. In other cases, a worker might be able to work on several tasks at the same time. In addition, the working capacity of a worker may dynamically change depending on his or her work schedule and on the nature of the tasks. For instance, a worker may be able to work on two simple tasks at the same time but can only work on one complex task.

Processing Time. The processing time of a task by a worker can be either deterministic or stochastic. In some cases, it is not unrealistic to assign a proficiency level to each worker and assume that the processing time of any task by a given worker is inversely proportional to his or her proficiency level. This is true when tasks are relatively homogeneous. In other cases involving more complex and heterogeneous tasks, however, there may exist high variance in the processing times needed for a worker to process different instances of the same type of task. Thus, the generic proficiency concept does not apply. Another important aspect of worker proficiency is the possible learning effect. A worker becomes more efficient when he or she cumulates experience with a particular type of task.

Our research focuses on a simplified workflow role resolution scenario. Related assumptions will be stated formally in §3. Here we summarize them in an informal manner in relation to the above discussion on role-resolution modeling. As for tasks, we assume neither deadlines nor dependencies among them. In addition, we assume that workers are the only relevant resource. In our model, tasks arrive dynamically but statistical patterns of task arrivals are unknown. In terms of worker modeling, we assume that every worker can work on only one task at a time. All processing times are deterministic and known a priori. Furthermore, we assume that workers are always available during working hours when the workflow system is managing and that no learning takes place.

These simplifying assumptions are made mainly because of the following considerations. First, the primary focus in this paper is dealing with dynamic task arrivals using online batching techniques. Simplifying along other dimensions of role resolution enables us to develop models of manageable complexity and produce concrete managerial insights. Second, support for role resolution from existing workflow technology is limited to scenarios where these simplifying assumptions hold (see §1.3). In effect, some of the assumptions, e.g., the one regarding task dependency, are not as limiting as they sound; see §6.1 for detailed discussion.

1.3. Workflow Role Resolution in Practice

Support for role resolution in current commercially available workflow systems can be categorized into three major types: system built-in policies, user-customizable policies, and rule-based policies.

Most workflow systems (e.g., Ultimus 5.0 and Staffware 7.0) provide some built-in role-resolution policies such as *a shared queue of tasks* and *load-balanced task assignment*. In the shared-queue approach, all tasks of the same type are placed in a common queue shared by all workers with the corresponding role. These qualified workers then select tasks to perform. In the load-balanced approach, the workflow system pushes a new task to an individual worker who has the right qualifications and the least amount of work.

Some of the workflow systems (e.g., Oracle Workflow 2.6 and Carnot), in addition to making available the above policies as built-in functions, also allow customized role-resolution policies. This type of workflow-management system offers opportunity to implement more advanced role-resolution policies beyond shared queues and load-balanced assignment.

In some application areas of workflow involving a large volume of tasks and a large number of workers, sophisticated task-allocation mechanisms such as rule-based approaches have been used (Bernett and Jaramillo 2001). For instance, in call-center operations, many systems (e.g., Pipkins and Avaya) have implemented rule-based task allocation that explicitly encodes domain-dependent matching conditions between tasks and workers.

Existing role-resolution policies such as a shared queue of tasks and load-balanced task assignment are easy to implement and understand. They are effective in cases where each worker has a limited set of active roles and where worker proficiencies for each role are relatively uniform. However, under multitasking environments, these assumptions do not hold, resulting in inefficient workflow operations. More advanced rule-based approaches are flexible in implementing specialized task-assignment policies and dealing with special business constraints. However, such approaches do not explicitly take into account operational effectiveness and could be further improved.

Some workflow systems are beginning to address the efficiency issue of role resolution by providing guidance to individual workers on task selection. For instance, the WorkMovr system uses a work-allocation technique called "work scoring" that ranks every task in the collection of work items by considering skill preference, role preference, and work-queue preference, etc. Although the models used in this system are still simplistic, they point out the need for more sophisticated role-resolution models and computational methods.

1.4. Research Objectives and Structure of the Paper

The objectives of our research are threefold. First, we formalize workflow role resolution decisions in an optimization-based framework and point out the relevance and technical differences between these role-resolution models and well-established assignment and scheduling models. Second, based on this framework, we develop several new role-resolution policies using batching-based online optimization techniques. Third, we evaluate various existing policies along with these new policies and compare their effectiveness and applicability in various workflow scenarios. We also explore issues with respect to implementation of these policies in commercial workflow systems.

The rest of the paper is structured as follows. Section 2 discusses the relevant literature. Section 3 formulates several role-resolution decision problems in an optimization framework and discusses related computational issues. We summarize in §4 current practice and propose several new online batching-based optimization policies. Section 5 presents a computational study to evaluate our proposed policies relative to those stemming from current workflow systems, and analyzes the simulation results. We conclude in §6 with a summary of our main research findings, including potential system and managerial implications of our research, and a discussion of future research directions.

2. Related Work

Coordination theory provides a conceptual framework to understand the impact of process management and its automation through workflow systems on organizational design and change. Malone and Crowston (1994) surveyed coordination theory as an emerging research area evolving from computer science, organization theory, operations research, economics, linguistics, and psychology. A key insight of their framework is that coordination can be seen as the process of managing dependencies among activities. They advocated that future research should develop more advanced coordination processes and cooperative work tools. Crowston (1997) further demonstrated that the analysis of exiting organizational processes and their rearrangements can affect the design of new organizational forms. Recently, Malone et al. (1999) described an online "process handbook," through which business process redesign and knowledge management can be facilitated. Our work reported in this paper can be viewed as an effective operational-level mechanism that embodies and implements some of the principles proposed and studied in coordination theory.

From modeling and computational perspectives, task and role designs are two research issues closely

related to workflow role resolution. A *task* is a collection of work steps grouped into a unit of work, and a *role* is description of qualification that can be performed by workers that possess the required skills. Dewan et al. (1997) presented a method of business process and task redesign by job consolidation. Their initial results revealed when the loss of specialization and lack of task control can undermine the benefits of cycle-time reduction through consolidation. Their work points out the importance of role design and the potential impact of role resolution.

There exists a distinct yet related body of literature focusing on modeling various aspects of the software-engineering process and studying how to accommodate and facilitate changes in different stages of information system design and implementation (Deruelle et al. 2001, Dhar and Jarke 1992). Research conducted in this area can be potentially generalized to provide an alternative formal framework and related computational methods (e.g., graph rewriting methods, Rajlich 1997) for task modeling, and change management when extensive interdependences among tasks are present.

Shen et al. (2003) proposed a task-assignment algorithm using a fuzzy-logic approach that considers the skills of workers, task complexity, and worker relationships. Although it remains unclear how worker relationships can be explicitly captured, their work argues convincingly for the importance of efficient task assignment. Their model assumed that each task instance is allocated independently of other instances of the same task type. We, on the other hand, consider realtime arrival of tasks and allocate them in a batching scheme.

Designing competent and efficient organizations is the foundation of workflow role resolution. At the minimum, effective workflow role resolution depends on availability of well-designed tasks and roles that are specified in sufficient details. Karageorgos et al. (2002) proposed a method for designing agent organizations in a semi-automated manner. Their proposed approach enables agent system designers to reason at a high abstraction level and conveniently reuse previous design decisions.

The assignment and scheduling literature provides the basic analytical framework to model role-resolution decisions (Baker 1974, Weng et al. 2001, Borodin and El-Yaniv 1998, Azar et al. 1995). Assignment models study the most cost-effective assignment of a given set of tasks to a given set of resources (e.g., people, machines), assuming that the cost of assigning each task to each resource is known. When the capacities of the resources are considered, assignment problems are called *generalized* assignment problems that are NP-hard (Racer and Amini 1994).

In assignment models, tasks are assumed to be independent of each other. Scheduling models study similar task-allocation decisions but consider more complex modeling of tasks (e.g., precedence constraints among tasks) and resources (e.g., setup costs) and make sequencing and timing decisions among and for tasks (Baker 1974, Morton and Pentico 1993, Herrmann et al. 1997). Recent years have seen a tremendous growth in the literature on stochastic scheduling where task information is only partially known and the average performance of the scheduling system is of the primary concern (Coffman and Whitt 1995).

Although the existing assignment and scheduling literature offers a wide array of models and computational methods (e.g., many of the modeling considerations discussed in §1.2 have been studied in the scheduling literature, Morton and Pentico 1993), which serve as a starting point for the study of role-resolution decisions, the specific characteristics of workflow management call for new research. For instance, in many workflow environments, tasks arrive dynamically and must be processed in a short time frame to assure quality of service. Therefore, waiting for tasks to accumulate into a large batch and then applying assignment or scheduling algorithms may lead to suboptimal performance. On the other hand, many tasks may take different workers substantially different amounts of time to finish. This high variance in processing times offers more opportunity to optimal task assignment and therefore favors an optimization approach based on relatively large batches. How to balance these two potentially conflicting considerations poses an interesting research question. In addition, workers, unlike machines, may have their own preferences as to which tasks they want to perform. How to make assignment and scheduling decisions with consideration of these preferences leads to new modeling and computational challenges.

3. Formal Models of Workflow Role Resolution

As stated in §1.1, we assume that staffing decisions are made prior to role-resolution decisions and are taken as input to our models. In this section, we formally develop various optimization-based models. These models help analyze how computationally extensive it is to make optimal role-resolution decisions in various settings and motivate online policies studied in the rest of the paper.

First we consider the static version of role resolution. There is a finite set of independent tasks T. All tasks are available at the beginning of the time period for immediate execution. There exists a finite set W of

workers who are relying on the workflow system for role resolution and task assignment. Assume worker i takes p_{ij} time units to complete task j and that the pay rate for worker i per time unit is w_i. Furthermore, we assume that each task can be processed by at most one worker, and each worker can handle at most one task at a time. Every task must be executed continuously from start to completion without interruption or restart.

The objective of cost minimization (the *CM problem*), measured by worker wages, can be formulated as follows:

$$\min \sum_{i\in W,\, j\in T} w_i p_{ij} x_{ij} \tag{1}$$

$$\text{subject to: } \sum_{i\in W} x_{ij} = 1 \quad \forall j \in T \tag{2}$$

$$x_{ij} = 0 \quad \text{or} \quad x_{ij} = 1 \quad \forall i \in W,\ \forall j \in T. \tag{3}$$

When $x_{ij} = 1$, task j is assigned to worker i, and (2) ensures that each task is assigned to exactly one worker.

We define the *flowtime* of a task as the time between the task's arrival and its completion (Baker 1974). The objective of minimizing the maximum task flowtime (the *MF problem*) from the given set of tasks can be formulated as follows:

$$\min \ z \tag{4}$$

$$\text{subject to: } \sum_{i\in W} x_{ij} = 1 \quad \forall j \in T \tag{5}$$

$$\sum_{j\in T} x_{ij} p_{ij} \le z \quad \forall i \in W \tag{6}$$

$$x_{ij} = 0 \quad \text{or} \quad x_{ij} = 1 \quad \forall i \in W, \forall j \in T. \tag{7}$$

This formulation differs from the CM formulation in using a different objective function and introducing constraint (6), ensuring that z equals the maximum flowtime across all tasks.

The prevailing *least loaded qualified person* (LLQP) policy—assigning the task to the least loaded qualified person (see §4 for a detailed discussion)—is optimal for both minimization problems above if p_{ij} and w_i are constant. This corresponds to business scenarios where all the workers are equally competent and all the tasks are of the same type. In most workflow applications, these assumptions do not hold. As such, the use of LLQP can lead to suboptimal solutions with increased cost and low quality of service.

We are mainly concerned with the MF problem and its variations which focus on reducing flow time (or equivalently, improving guaranteed quality of service). When p_{ij} are not constants and vary between tasks and workers, the MF problem becomes hard to solve. In general, it is an NP-hard problem in the strong sense (Garey and Johnson 1979). Even for a special case where all workers are identical, i.e., $p_{ij} = p_{i'j}$ for any $i \in W$, $i' \in W$, and $j \in T$, which has been extensively studied in the operations research and computer science literature as the parallel-processor machine scheduling problem, the problem is still NP-hard in the strong sense.

We now consider the dynamic version of role resolution where not all of the tasks are available at the beginning of the time period for execution. Rather, these tasks arrive dynamically. Let r_j be the time when task j arrives. As in the static version of the problem, there exists a finite set W of workers who rely on the workflow system for task assignment, and task j takes p_{ij} time units for worker i to complete it. Note that we assume that processing times are deterministic.

We extend the MF formulation by incorporating dynamic arrival of tasks as follows (referred to as the *DMF problem*):

$$\min \ z \tag{8}$$

$$\text{subject to: } \sum_{i\in W}\sum_{k\in T} x_{ijk} = 1 \quad \forall j \in T \tag{9}$$

$$s_j \ge r_j \quad \forall j \in T \tag{10}$$

$$(x_{ijk} + x_{ij'(k+1)} - 1)(s_j + p_{ij}) \le s_{j'} \quad \forall i \in W,\ \forall k \in T,\quad \forall j \in T,\ \forall j' \in T \tag{11}$$

$$s_j + \sum_{i\in W}\sum_{k\in T} p_{ij} x_{ijk} - r_j \le z \quad \forall j \in T \tag{12}$$

$$x_{ijk} = 0 \quad \text{or} \quad x_{ijk} = 1 \quad \forall i \in W,\ \forall j \in T,\ \forall k \in T \tag{13}$$

$$s_j \ge 0. \tag{14}$$

When $x_{ijk} = 1$, task j is assigned to worker i as the kth task. The index k indicates sequencing of tasks for each worker. Variable s_j indicates the starting time for task j;

(9) ensures that each task is assigned to exactly one worker;

(10) guarantees that a task does not start before it arrives;

(11) ensures that one worker handles at most one task at any given time;

(12) ensures z equals to the maximum flowtime across all tasks.

The above formulation contains nonlinear constraints in (11). By introducing some auxiliary variables, we can convert the formulation into a mixed integer program. The DMF problem is obviously NP-hard in the strong sense.

Applying the DMF model in practice poses two problems. First, complete information regarding task arrivals and processing times is needed to derive optimal decisions. In practice, such information is

unknown prior to actual task arrivals. Second, even if such information is known (or can be estimated accurately), the model requires extensive computational resources to solve (Garey and Johnson 1979). Nonetheless, this model is useful in two main ways. First, it provides a theoretical reference based on which role-resolution policies can be evaluated. Second, although the DMF model cannot be directly applied to optimize role-resolution decisions, it can be used in a localized online context and help derive several new role-resolution policies described in §4.

4. Workflow Role-Resolution Policies

In this section, we investigate five role-resolution policies. The first two are based on policies currently used in commercial workflow systems. The last three are new policies that we develop to take advantage of batching-based online optimization.

1. The first policy is a load-balanced approach. An incoming task is assigned, as soon as it arrives, to a qualified worker whose task queue is among the shortest. The worker will execute the tasks one by one from his/her individual queue on a first-in-first-out (FIFO) basis. This is the *LLQP policy*. This policy can be characterized as the simplest *push* policy.

2. The second policy is to maintain a single queue shared by all qualified workers who execute tasks in this shared queue on a FIFO basis. No attempt is made in this policy to assign tasks optimally. This approach is referred to as the *shared queue* (SQ) policy.

3. The third policy considers optimal task assignment by maintaining a shared queue and an individual queue for each worker. The transfer of tasks from the shared queue to the individual queue is done using an optimal task-allocation procedure when the shared queue reaches a batch size of K, where K is chosen heuristically by the policy designer. This approach is referred to as the K-*Batch policy*.

4. The fourth policy modifies the third one by reducing the individual queue size to one. That is, while the optimal assignment procedure is computed when the shared queue reaches a batch size of K, the resulting optimal assignment is not fully committed to the workers. Instead, only the workers whose individual queue becomes empty receive the first task assigned to them. The other uncommitted tasks will remain in the shared queue and be reassigned in the next run. This approach is referred to as the K-*Batch-1 policy*.

5. The fifth policy modifies the fourth one by relaxing the batch-size constraint. That is, the optimal task-assignment procedure is run as soon as any worker's task queue becomes empty. This approach is referred to as the *1-Batch-1 policy*.

Under K-Batch, K-Batch-1, and 1-Batch-1, an optimization problem must be solved to assign tasks. This problem is formulated as that of minimizing the maximum flowtime given the dynamic availability of the workers. We call this problem the *minimizing sequential assignment* (*MSA*) problem. Let a_i denote the time when worker i will finish all previously committed tasks and thus be available. The MSA problem can then be formulated as:

$$\min z \tag{15}$$

subject to:

$$\sum_{i\in W} x_{ij} = 1 \quad \forall j \in T \tag{16}$$

$$a_i + \sum_{j\in T} x_{ij} p_{ij} \le z \quad \forall i \in W \tag{17}$$

$$x_{ij} = 0 \quad \text{or} \quad x_{ij} = 1 \quad \forall i \in W,\ \forall j \in T. \tag{18}$$

The MSA model differs from both the MF model and the DMF model. MSA is more general than MF because in MF all workers are available at time zero while MSA factors a_i into the flowtime computation (17). MSA greatly simplifies DMF in that only the tasks immediately available for execution are considered. Computationally, unlike DMF, MSA is amenable to exact optimization techniques for small problems, although it is still NP-hard in the strong sense. In the computational study reported in the next section, MSA problem instances were solved optimally using a mixed integer program solver. We have also developed several heuristic methods to solve large MSA problems efficiently. For purposes of comparing given role-resolution policies, these heuristic methods yield results very close to those from the exact optimal method. Due to limited space, we do not discuss these MSA heuristics in this paper.

Under K-Batch-1 and 1-Batch-1, certain tasks could be kept in the shared queue without being assigned to individual workers. We use the following heuristic approach to avoid this potential problem: for setting MSA model parameters, a task's processing times are artificially shortened exponentially depending on the length of the time this task stays in the share queue. For instance, when a task enters the shared queue, its actual processing times are fed to the MSA model to make assignment decisions. If this task stays in the queue for n time units, future invocations of the MSA model will use α^n times its actual processing times as processing-time parameters for this task; we set $\alpha = 0.95$.

5. A Computational Study

Under the incomplete-information assumption regarding task arrivals, a meaningful analytical evaluation of the five role-resolution policies under study

Table 1 Simulation Parameters

Number of workers	5, 10
Average system load	0.2 (low), 0.6 (medium), 1.0 (high)
Task variability (%)	20 (low), 80 (medium), 140 (high)
Worker variability (%)	20 (low), 80 (medium), 140 (high)

is unlikely. (Some coarse comparison is possible using online algorithm analysis techniques; see Borodin and El-Yaniv 1998. Such a comparison is beyond the scope of this paper.) Therefore, we have conducted a computational experiment to compare the performance of the five policies, i.e., LLPQ, SQ, K-Batch, K-Batch-1, and 1-Batch-1. In this section, we describe in detail our simulation setup and report our initial findings based on simulation results.

5.1. Problem-Instance Generation

Our computational study is mainly concerned with evaluating the performance of the five policies under study in various business operational environments. The parameters used to characterize the operational scenarios and their specific settings examined in our study are summarized in Table 1. The values of the parameters used in our simulation are based on real-world workflow management scenarios (Bernett and Jaramillo 2001).

We chose relatively small values (5 and 10) for the number of workers because in a typical workflow application setting, it is common that only a small number of workers are assigned to a given role. Consequently, our simulation program uses the mixed integer program (MIP) solver CPLEX to make MSA decisions in an optimal manner for K-Batch, K-Batch-1, and 1-Batch-1.

The average system load measures the task arrival rate relative to the aggregate worker capacity. We differentiate between individual workload and system load. The workload on individual workers is not part of the independent variables characterizing the business operational environment. In effect, individual workload is a *dependent* variable and was used in our study as an evaluation criterion for a chosen policy. The system load, however, is an independent variable introduced in our simulation to control the dynamic generation of tasks.

The detailed procedure used to generate tasks is as follows. Without loss of generality, we fixed the average task processing time to 10 units of time in our simulation, and therefore the average worker capacity per time unit is 1/10. Assuming that n workers are present in the system, the total available worker capacity per time unit is $n/10$. When the average system load is set to l, the number of tasks that arrive each time unit is assumed to follow a Poisson distribution with $\lambda = ln/10$. For instance, when $n = 10$ and $l = 0.6$, the number of (new) tasks that are generated for each time unit follows a Poisson distribution with $\lambda = 0.6$. The values of l were chosen to represent three operational scenarios: light load (0.2), medium load (0.6), and high load (1.0).

The two remaining parameters, task variability and worker variability, govern the generation of task processing time. In our study, p_{ij} are generated in a two-step process.

Step 1. For task j, the average processing time $\overline{p_j}$ across workers is generated based on the overall average task processing time (which was set to 10) and task variability *varTask* based on normal distribution $N(10, (10 \times varTask)^2)$. In other words, *varTask* is the standard deviation relative to the mean processing time.

Step 2. p_{ij} is generated after $\overline{p_j}$ is obtained. Defined similarly to task variability, worker variability *varWorker* indicates the percentage deviation from the mean individual task processing time. As such, p_{ij} is generated according to the normal distribution $N(\overline{p_j}, (\overline{p_j} \times varWorker)^2)$.

In our study, we rounded all the time measures to integer values and negative realizations were reset to one. For both task variability and worker variability, we experimented with the following value settings: 20% representing low variability, 80% medium variability, and 140% high variability.

5.2. Simulation Runs

Table 1 represents a total of $2 \times 3 \times 3 \times 3 = 54$ experimental conditions. For each condition, we randomly generated 20 problem instances and then ran all five policies to solve them. In total, we simulated 5,400 problem instances. We posed a limit on the number of dynamically generated tasks to terminate simulation runs.

The simulation program was written using a combination of the PYTHON scripting language for the main control flow of simulation and data collection, and AMPL/CPLEX for solving the MSA problem optimally under K-Batch, K-Batch-1, and 1-Batch-1 policies. The batch size K can affect the performance of K-Batch and K-Batch-1 policies. Large K indicates longer batching time between consecutive batches. While longer batching time can lead to worker idleness and potentially worsened flowtime, it offers more complete information, and, as a result, potentially better optimization results. We have experimented with different values of K but only report in this paper simulation results where K was fixed at 5 due to limited space.

For each run, three key system performance measures were collected.

1. Maximum Flowtime. This objective is the main focus of our analysis. In business terms, maximum flowtime works toward guaranteed response time of all tasks, indicating the quality of service.

2. Average Workload. This measure indicates on average how busy the workers are under a given policy. For each individual worker, workload is calculated as the ratio between the total number of working time units and the overall length of the simulation. Managerially, average workload is a good indication of direct labor cost. In outsourcing practice, workload is the key deciding factor of cost. In regular in-house operations, wages are paid even when workers are idle. In this case, average workload provides important statistics in making staffing decisions.

3. Workload Variation. This measure is defined as the sample standard deviation of individual workloads. The managerial interpretation of this measure varies with business scenario. For instance, in in-house operations, workload variation provides a good measure of how "fairly" a policy is treating all employees.

5.3. Impact of Processing-Time Variations

In this section, we report our simulation results regarding the impacts of variation in task processing times on the three system performance measures, i.e., maximum flowtime, average workload, and workload variation. For ease of exposition, we focus mainly on the cases where the overall system load is light (20%). Note that in our simulation, there are two independent parameters, i.e., task variability and worker variability, each of which have three settings (low (20%), medium (80%), and high (140%)), resulting in nine combinations. However, we will show only four representative combinations out of the nine, namely $(varTask, varWorker) \in \{$(Low, Low), (Low, High), (Medium, Medium), (High, High)$\}$, which suffices for our purposes. Further, each data point in the result is the average of 20 instances with the same system load and processing time variability. In §5.4, we examine how the role-resolution policies behave under different overall system loads.

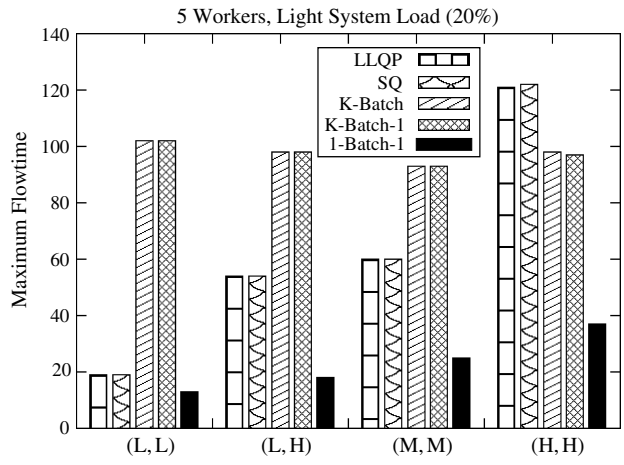


Figure 2 **Impact of Processing Time Variation on Maximum Flowtime**

Table 2 **Policy Comparison by Maximum Flowtime with Five Workers and Light System Load at** (L, L)

	SQ	K-Batch	K-Batch-1	1-Batch-1
LLQP	≈	<	<	>
SQ		<	<	>
K-Batch			≈	>
K-Batch-1				>

Figure 2 summarizes the relative performance of the five policies as measured by the maximum flowtime under varying degrees of processing-time variation. When the variation is low at (L, L), LLQP, SQ, and 1-Batch-1 policies offer significantly shortened maximum flowtime than do K-Batch and K-Batch-1. This indicates that the penalty of batching outweighs the potential gain in batched optimization. However, 1-Batch-1 does not suffer from the rigid batching scheme and therefore adapts to low-load cases. The above observations are confirmed by a series of statistical tests. For instance, Table 2 summarizes all pairwise comparisons between the five policies with five workers, light system load, and low task and worker variabilities at (L, L), using a paired t-test at the 0.01 significance level. In this table, "≈" indicates that there is no significant difference between the corresponding row and column policy with respect to maximum flowtime, while ">" ("<") indicates that the row policy results in significantly larger (smaller) maximum flowtime than the column policy. We also performed additional distribution-free tests as well as ANOVA with follow-up tests that control the experimentwise error rate (e.g., the Newman-Keuls test, Daniel 1990). These additional tests produce almost identical conclusions. A similar multimethod procedural was followed to confirm the experimental findings presented in the remainder of this section. For brevity, we summarize only a small, selected set of such statistical tests.

When processing-time variation increases, as illustrated at (L,H) and (M,M), the performance of K-Batch and K-Batch-1 improves relative to LLQP, SQ, and 1-Batch-1 on flowtime. This improvement can be explained by the fact that higher variation increases the potential benefit of optimization.

Table 3 **Policy Comparison by Maximum Flowtime with Five Workers and Light System Load at** (H, H)

	SQ	K-Batch	K-Batch-1	1-Batch-1
LLQP	≈	>	>	>
SQ		>	>	>
K-Batch			≈	>
K-Batch-1				>

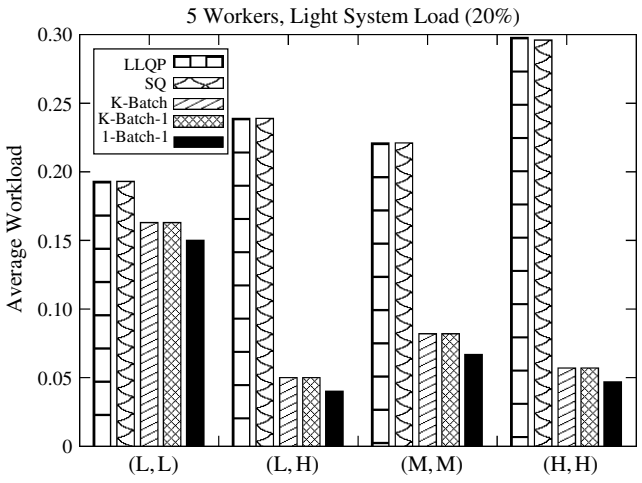


Figure 3 Impact of Processing-Time Variation on Average Workload

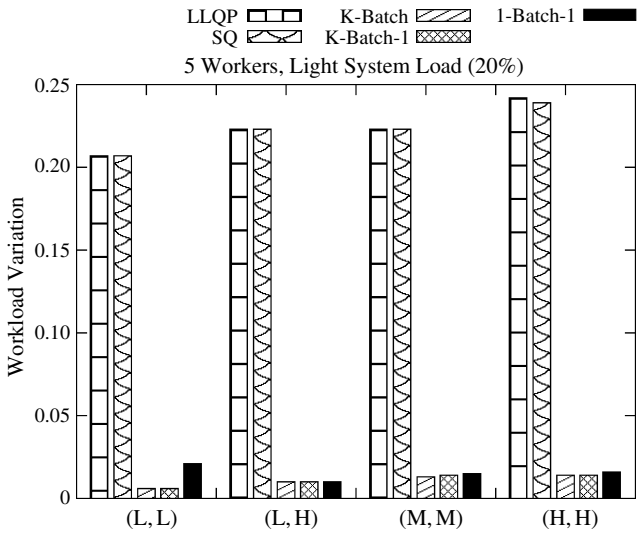


Figure 4 Impact of Processing-Time Variation on Workload Variation

At the high end of processing-time variation, as illustrated by (H, H), K-Batch and K-Batch-1 outperform LLQP and SQ on flowtime. In all the cases shown, 1-Batch-1 dominates all other policies on flowtime. Table 3 summarizes the results of the statistical tests performed to compare the five policies with five workers, light system load, and high task and worker variabilities.

Figure 3 summarizes the relative performance of these policies as measured by the average workload under different processing-time variation. As shown, optimization-based policies consistently deliver better results. This becomes increasingly evident when processing-time variation increases. Table 4 summarizes the statistical tests designed to compare the five policies with respect to average workload with five workers, light system load, and high task and worker variabilities.

Figure 4 compares the policies with respect to the resulting workload variation. As shown, optimization-based policies deliver significantly better results in terms of variation among individual workloads. Within optimization-based policies, K-Batch and K-Batch-1 seem to deliver better results than does 1-Batch-1. Table 5 summarizes the statistical tests performed to compare the five policies with respect to workload variation with five workers, light system load, and high task and worker variabilities.

Our computational results also show that these observations hold when the number of the workers is increased from 5 to 10. Figure 5 summarizes the performance of the five policies measured by maximum flowtime. It is interesting to notice that when the number of workers increases, optimization-based policies perform better relative to other policies. We omitted figures on average workload and workload variation for ten workers because they are similar to those for five workers.

5.4. Impact of System Load

In this section, we examine the impact of system load on flowtime performance of the role-resolution policies. As shown, the optimization-based policies, i.e., K-Batch, K-Batch-1, and 1-Batch-1, perform better at higher system load, and LLQP and SQ are favored at low system load.

Figure 6 illustrates the impact of system load on maximum flowtime with low processing-time variation. As shown, the performance of K- Batch and K-Batch-1 improves significantly when the system load increases. Under low and medium system load, LLQP and SQ outperform K-Batch and K-Batch-1. The 1-Batch-1 policy dominates other policies except in high system load. Table 6 summarizes statistical differences between the five policies under medium system load.

Figure 7 illustrates the impact of system load on maximum flowtime with medium processing-time

Table 4 Policy Comparison by Average Workload with Five Workers and Light System Load at (H, H)

	SQ	K-Batch	K-Batch-1	1-Batch-1
LLQP	≈	>	>	>
SQ		>	>	>
K-Batch			≈	>
K-Batch-1				>

Table 5 Policy Comparison by Workload Variation with Five Workers and Light System Load at (H, H)

	SQ	K-Batch	K-Batch-1	1-Batch-1
LLQP	≈	>	>	>
SQ		>	>	>
K-Batch			≈	≈
K-Batch-1				≈

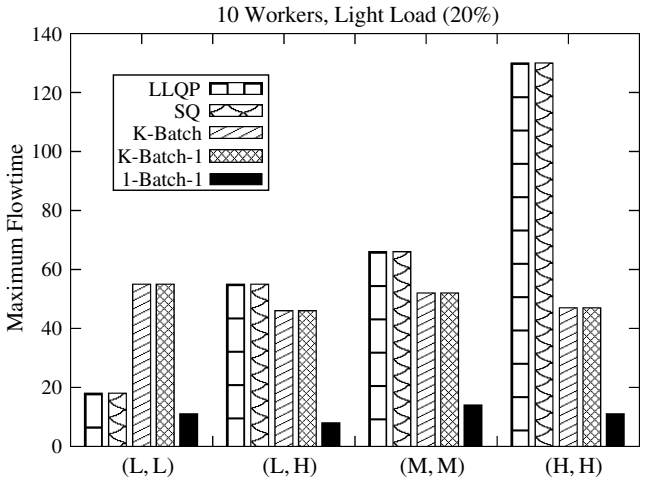


Figure 5 **Impact of Processing-Time Variation on Maximum Flowtime (10 Workers)**

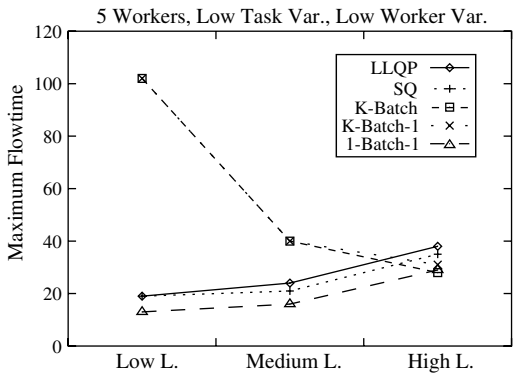


Figure 6 **Impact of System Load on Maximum Flowtime at Low Processing-Time Variation**

Table 6 **Policy Comparison by Maximum Flowtime with Five Workers and Medium System Load at** (L, L)

	SQ	K-Batch	K-Batch-1	1-Batch-1
LLQP	>	<	<	>
SQ		<	<	>
K-Batch			≈	>
K-Batch-1				>

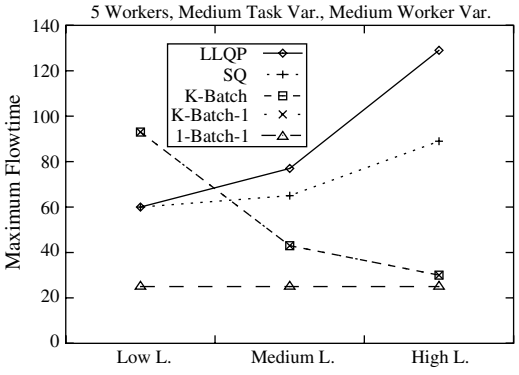


Figure 7 **Impact of System Load on Maximum Flowtime at Medium Processing-Time Variation**

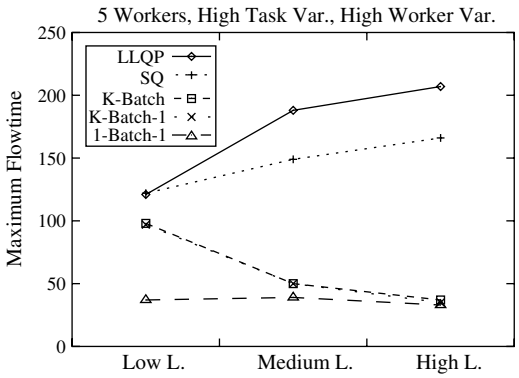


Figure 8 **Impact of System Load on Maximum Flowtime at High Processing-Time Variation**

variation. LLQP and SQ outperform K-Batch and K-Batch-1 when the system load is low, but this reverses as the system load turns to medium and high. The 1-Batch-1 policy outperforms the other four consistently.

Figure 8 illustrates the impact of system load on maximum flowtime with high processing-time variation. LLQP and SQ are inferior under all levels of system load, and the difference becomes more significant as the system load becomes higher. The 1-Batch-1 policy appears to be the best among the three optimization-based policies.

5.5. Summary of Key Findings

The key insights from this computational study are:

The workflow designer should consider using batching-based online optimization policies when the system load is medium to high.

Processing-time variation has a major impact on system performance, and higher variation favors optimization-based policies.

Average workload and workload variation can be significantly reduced using online optimization.

The online optimization policy without a fixed batch size (1-Batch-1) seems to consistently deliver the best performance under most operational conditions.

6. Conclusions and Future Work

In this paper, we have studied role-resolution policies in workflow systems. Role resolution is the key step in assigning dynamically arriving tasks to individual workers. It affects the quality of service and worker productivity. Beyond operational considerations, role resolution is also an integral part of strategic-level decisions including organizational job design and staff planning.

We proposed three online optimization-based role-resolution policies and studied their performance via simulation along with two other policies reflecting current workflow practice. We are primarily concerned with the following workflow system perfor-

mance measures: maximum flowtime, average workload, and workload variation. The impact of system load, task processing-time variation, and the number of workers on these performance measures was evaluated under the five policies examined.

We conclude the paper by discussing potential system and managerial implications of our work, and future research directions.

6.1. Systems and Managerial Implications

We now briefly discuss how the proposed role-resolution models and associated policies can be integrated into a workflow system.

As shown in Figure 9, the advanced role-resolution module can be developed as an extension to the standard workflow system architecture (Stohr and Zhao 2001). Under this extended architecture, a workflow engine calls upon the advanced role-resolution module when tasks need to be assigned. Depending on the nature of the role-resolution policy (e.g., shared queues versus individual queues), tasks are aggregated (batched) by the role-resolution module and then pushed to (or pulled by) workers either optimally or heuristically.

In applications where tasks are independent of each other (e.g., in many call-center operations, each incoming request is treated as a separate task even if the caller is following up on a previously reported problem), all the role-resolution policies discussed are directly applicable. In other applications where complex dependencies exist between the tasks (e.g., as in the claim-handling workflow illustrated in Figure 1), the role-resolution policies developed in this paper are still applicable. Operation of the role-resolution module now mimics the "myopic" practice widely accepted in scheduling theory and practice (Morton and Pentico 1993). At each decision making point, out of all tasks that need to be processed, the role-resolution module assigns only those tasks that can be *readily* processed and ignores other tasks that are not immediately available for execution (e.g., because their predecessor tasks are not yet finished). This approach is of a myopic nature because it makes task-assignment decisions based on what can be immediately done rather than on a more global view including information concerning future tasks. Mathematically, this myopic approach can lead to suboptimal solutions. However, it helps to reduce greatly the complexity of decision making and in many cases delivers near-optimal solutions; an alternative would be an unmanageable dynamic project-scheduling problem (Morton and Pentico 1993). We thus conclude that in principle our approach can be applied for general workflow management.

Effective operation of the proposed approach hinges on the availability of accurate estimates of the processing time of each worker for every task type.

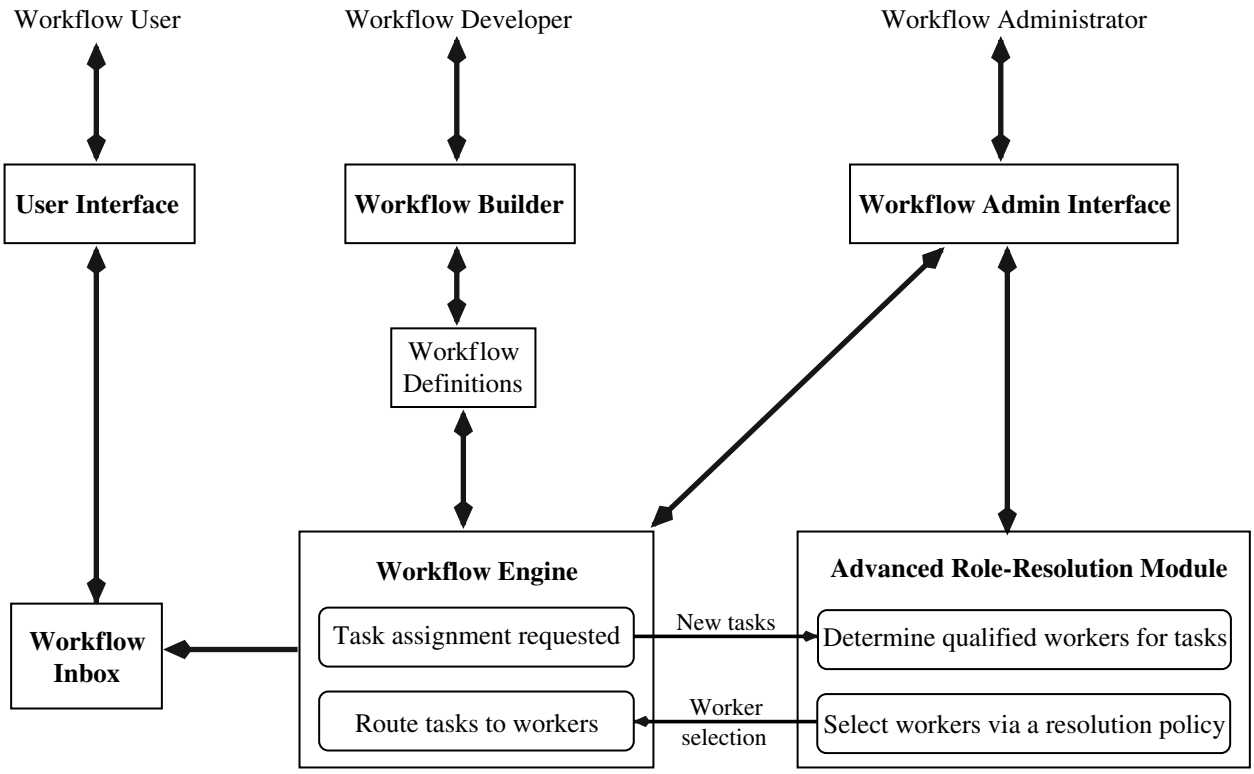


Figure 9 Workflow Architecture with Advanced Role Resolution

We envision that this problem can be solved using a mix of approaches. During the initial operation of the system, workflow administrators can generate initial estimates. When the system accumulates sufficient task logs, statistical methods can be employed to derive more accurate estimates.

6.2. Future Research Directions

We are currently extending our work to deal with issues that can arise when implementing our approach in real business settings.

Deploying the proposed optimization-based policies could lead to the following two practical implementation issues. (a) These optimization-based policies are largely a push-based approach that does not explicitly consider worker preferences and initiatives. In general, it is more likely that workers will perceive pull-based approaches to be more "user friendly" and appealing where the workers themselves play a significant role in deciding what tasks will be assigned to them. (b) Overspecialization could occur when the proposed policies are applied rigidly. Under these optimization-based policies, a worker who is efficient at one type of task tends to get assigned similar tasks and not other types of tasks. While over-specialization may not present a problem in some applications, it may cause managerial concerns in other situations.

The over-specialization issue can be relatively easily dealt with by pull-based approaches. However, our initial analysis indicates that simple pull-based approaches based on simplistic worker preferences (e.g., a worker prefers a task that will take him the least amount of time to finish among the given tasks) result in significantly worsened workflow performance than a push approach such as 1-Batch-1. We are currently developing more elaborate and autonomous models of workers and designing hybrid role-resolution policies that take advantages of both push- and pull-based approaches. More specifically, we are examining "soft"-push approaches where workers are allowed to "pull" tasks from a ranked task list. Such ranking will be produced by optimization-based policies. In addition, we are studying policies based on distributed computational mechanisms such as auctions (Bertsekas et al. 1992, Tan and Harker 1999).

Job design and staffing are another interesting area of research. Current workflow literature assumes that the job design and staffing decisions are given. This assumption in many cases is not realistic and often renders the workflow design ineffective. Given the proliferation of workflow applications, future organizations could be designed more rigorously using decision support systems. This will require knowledge of the available workforce, worker skills, multifunctionality policy, and employee-development plan, among others. Our future work will study job design and staffing models and explore ways through which these models can be effectively integrated with role-resolution models.

Acknowledgments

The authors appreciate the comments of the anonymous reviewers. They also thank attendees of a University of Minnesota Information and Decision Sciences Workshop for their constructive suggestions. The first author is also affiliated with The Key Lab of Complex Systems and Intelligence Science, Chinese Academy of Sciences (CAS), Beijing, and was supported in part by a grant for open research projects (ORP-0303) from CAS.

References

Aissi, S., P. Malu, K. Srinivasan. 2002. E-business process modeling: The next big step. *IEEE Comput.* **35** 55–62.

Azar, Y., J. Naor, R. Rom. 1995. The competitiveness of on-line assignments. *J. Algorithms* **18** 221–237.

Baker, K. R. 1974. *Introduction to Sequencing and Scheduling.* Wiley, New York.

Bernett, H., M. Jaramillo. 2001. Assessing web-enabled call center technologies. *IEEE IT Pro* **3** 24–30.

Bertsekas, D. P., S. Pallottino, M. G. Scutello. 1992. Polynomial auction algorithms for shortest paths. Technical Report CICS-P-335, Center for Intelligent Control Systems, MIT, Cambridge, MA.

Borodin, A., R. El-Yaniv. 1998. *Online Computation and Competitive Analysis.* Cambridge University Press, Cambridge, UK.

Cheng, E. C. 2000. An object-oriented organizational model to support dynamic role-based access control in electronic commerce. *Decision Support Systems* **29** 357–369.

Coffman, E. G., W. Whitt. 1995. Recent asymptotic results in the probabilistic analysis of schedule makespans. P. Chrétienne, E. G. Coffman Jr., J. K. Lenstra, Z. Liu, eds. *Scheduling Theory and its Applications.* Wiley, New York, 15–31.

Crowston, K. 1997. A coordination theory approach to organizational process design. *Organ. Sci.* **8** 157–175.

Daniel, W. 1990. *Applied Nonparametric Statistics*, 2nd ed. PWS-Kent, Inc., Boston, MA.

Deruelle, L., M. Bouneffa, N. Melab, H. Basson. 2001. A change propagation model and platform for multi-database applications. *IEEE Internat. Conf. Software Maintenance* (*ICSM'01*), Florence, Italy, 42–51.

Dewan, R., A. Seidmann, Z. Walter. 1997. Workflow redesign through consolidation in information-intensive business processes. *The Eighteenth Internat. Conf. Inform. Systems.* Atlanta, GA, 285–296.

Dhar, V., M. Jarke. 1992. Conceptual modeling and change propagation. E. Stohr, B. Konsynski, eds. *Information Systems and Decision Processes.* IEEE Computer Society Press, Los Alamitos, CA, 217–230.

Garey, M., D. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York.

Herrmann, J., J.-M. Proth, N. Sauer. 1997. Heuristics for unrelated machine scheduling with precedence constraints. *Eur. J. Oper. Res.* **102** 528–537.

Karageorgos, A., S. Thompson, N. Mehandjiev. 2002. Semi-automatic design of agent organisations. *The Seventeenth ACM Symp. Appl. Comput.* Madrid, Spain, 306–313.

Malone, T., K. Crowston. 1994. The interdisciplinary study of coordination. *ACM Computing Surveys* **26** 87–119.

Malone, T., K. Crowston, J. Lee, B. Pentland, C. Dellarocas, G. Wyner, J. Quimby, C. Osborn, A. Bernstein, G. Herman, M. Klein, E. O'Donnell. 1999. Tools for inventing organizations: Toward a handbook of organizational processes. *Management Sci.* **45** 425–443.

Morton, T. E., D. W. Pentico. 1993. *Heuristic Scheduling Systems: With Application to Production Systems and Product Management.* Wiley, New York.

Pinker, E. J., A. Seidmann, R. C. Foster. 2002. Strategies for transitioning "old economy" firms to e-business. *Comm. ACM* **45** 77–83.

Racer, M., M. M. Amini. 1994. A robust heuristic for the generalized assignment problem. *Ann. Oper. Res.* **50** 487–503.

Rajlich, V. 1997. A model for change propagation based on graph rewriting. *IEEE Internat. Conf. Software Maintenance*, Bari, Italy, 84–91.

Shen, M., G. H. Tzen, D. R. Liu. 2003. Multi-criteria task assignment in workflow management systems. *Proc. Thirtysixth Hawaii Internat. Conf. System Sci.*, Big Island, Hawaii, 202.

Stohr, E. A., J. L. Zhao. 2001. Workflow automation: Overview and research issues. *Inform. Systems Frontiers* **3** 281–296.

Tan, J., P. Harker. 1999. Designing workflow coordination: Centralized versus market-based mechanisms. *Inform. Systems Res.* **10** 328–342.

van den Beukel, A., E. Molleman. 2002. Too little, too much—downsides of multifunctionality in team-based work. *Personnel Rev.* **31** 482–494.

Weng, M., J. Lu, H. Ren. 2001. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *Internat. J. Production Econom.* **70** 215–226.

Zhao, J. L., E. A. Stohr. 1999. Temporal workflow management in a claim handling system. *Software Engrg. Notes* **24** 187–195.