CrossMark

Building Automated Data Driven Systems for IT Service Management

Yixin Diao[1] · **Larisa Shwartz**[1]

Abstract Enterprises and service providers are increasingly challenged with improving the quality of service delivery while containing the cost. However, it is often difficult to effectively manage the complex relationships among dynamic customer workloads, strict service level requirements, and efficient service management processes. In this paper, we present our progress on building autonomic systems for IT service management through a collection of automated data driven methodologies. This includes the design of feedback controllers for workload management, the use of simulation-optimization methodology for workforce management, and the development of machine learning models for event management. We demonstrate the applicability of the presented approaches using examples and data from a large IT service delivery environment.

Keywords IT service management · Autonomic computing · Feedback control · Simulation-based optimization · Recommender system

1 Introduction

In recent years, the Information Technology (IT) service management industry has faced continual pressure to improve service quality while simultaneously reducing service delivery and management costs. These objectives are particularly critical for IT service outsourcing, in which service providers manage the IT infrastructures, applications, and business processes on behalf of their customers. To meet the needs of the customers at reduced prices and an accelerated time to market, service providers must efficiently tackle the scale and complexity of a managed IT

✉ Yixin Diao
diao@us.ibm.com

1   IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

environment as well as the diversity of processes across all of their outsourcing customers.

In contrast to IT infrastructure management where the focus is on IT systems such as cloud, networks, databases, and applications, IT service management is concerned with the operational aspects of IT management and adopts a more business and customer oriented approach to delivering IT services. The relationship between IT infrastructure management and IT service management is illustrated in Fig. 1. The left hand side indicates IT infrastructure and applications. The right hand side indicates IT service management, which is labor intensive and comprises a set of management processes and technologies that enables service providers to manage the IT infrastructure and applications from the perspective of their customers and of their own business.

The most adopted framework for IT Service Management is ITIL (Information Technology Infrastructure Library) [1], which systematizes the planning, delivery, and support of IT services. More specifically, ITIL adopts a five-stage service lifecycle comprising Service Strategy, Service Design, Service Transition, Service Operation, and Continuous Service Improvement. Each of them is organized by a set of service management processes. The focus of *Service Strategy* is to understand customer needs in order to provide the right mix of services to meet required business outcomes. *Service Design* is aimed to design IT services to ensure high quality service delivery and cost effective service provision. Afterwards, *Service Transition* develops and transits the new or transformed services into steady-state service delivery. Once at the steady-state, *Service Operation* is focused on achieving effectiveness and efficiency in service delivery to ensure value for both customers and service providers. Finally, *Continuous Service Improvement* emphasizes creating and maintaining value for customers through better design and operation.

In a global service delivery context, the service provider manages global customers' outsourced IT Services such as operating networks, supporting databases, and providing backup and restore services. The infrastructure supporting the services may be owned by the customers and located on customer sites; alternatively, it may be owned and located on provider sites on behalf of the customers. As service requests arrive from the customers, the requests are assigned
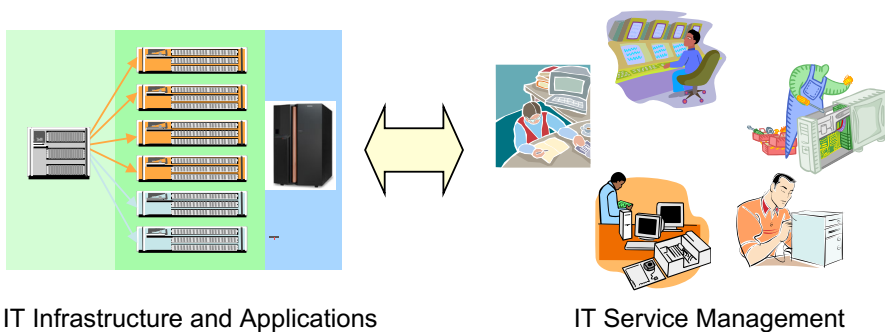

IT Infrastructure and Applications   IT Service Management

Fig. 1 IT service management

to service agents in various service delivery centers that are also globally located to both leverage qualified local skills and to provide round-the-clock support. On the other hand, however, the global service provider needs to manage hundreds of delivery locations, thousands of customers, and millions of service tasks. Such an enormous complexity challenges the service provider to ensure that processes are consistent across the whole services delivery environment and that high quality services are uniformly provided to all service customers by all service agent teams.

During the execution of IT service management processes, IT service providers track customer interactions and IT system performance, producing large volumes of data. Such data present the opportunities for data driven analytics to produce valuable insights and management actions for automated management of IT service processes. Building automated systems for IT service management shares some similar characteristics as building autonomic systems for IT infrastructure management: operating self-managed systems based on high-level objectives from the service owners, embracing a rigorous and model-based MAPE-K loop to improve management efficiency and quality, and freeing service providers from the burden of dealing with many low-level, yet vital, operational functions. On the other hand, since IT service management is inherently a human management process, human, as an important element, will be present both in the managed system and in the managing system. Indeed, the objective of automated service management systems is not to remove the human out of the loop, but to remove the ad-hoc process out of the loop.

There are several benefits for building automated data driven systems for IT service management. First, it lowers the complexity to integrate multiple heterogeneous environments into customer support service systems. Second, it enhances the IT service management best practices and processes by providing automated solutions. Third, it reduces the level of skills needed to design, configure, optimize, and maintain IT services. Finally, it improves consistency and repeatability across different organizations and for different customers. On the other hand, the challenges of building automated systems for IT service management lie in the needs to manage high uncertainties from complex tasks and to deal with massive data with large inaccuracies.

This paper summarizes the progress that we have made on building automated data driven service management systems. We start from a relatively simple example to dynamically adjust the size of the service delivery team in response to workload changes. We achieve this by developing a feedback control mechanism and the objective is to be able to have reasonable management performance given the high uncertainties encountered in the service delivery data. Based on the initial success, we further adopt the simulation-based optimization methodology to fully comprehend the different levels of complexities in service management. This involves a discrete event simulation model that captures the relationships among dynamic customer workloads, strict service level constraints, and service personnel with diverse skill sets, as well as an optimization model to provide recommended staffing levels with respect to the required skills and shift schedules. Finally, we discuss the use of machine learning techniques to build an automated recommender system for event management. This includes the extension of the k-nearest neighbors algorithm

in order to manage the complexities of understanding unstructured service data that are text heavy.

The remainder of this paper is organized as follows. Section 2 discusses the design of the feedback controller for workload management. Section 3 presents the simulation-optimization methodology for workforce management. Section 4 presents the use of machine learning models for event management. Section 5 reviews the related work. Our conclusions and future work are contained in Sect. 6.

2 Feedback Control for Workload Management

In this section, we describe an automated approach for workload management [2]. This supports the *Incident Management* process within ITIL's *Service Operation* lifecycle. The presented approach employs the feedback control methodology to dynamically adjust the organization of service delivery personnel based upon the observed gaps between measured and target service level metrics in response to the changes in the business environment. Compared to existing feedback control approaches used in systems management, the presented approach differentiates itself by (1) devising a means to decompose the combined service level optimization problem into a set of single-input single-output control problems with specific service operation targets as reference inputs, and (2) designing an uncertainty-based adaptive control approach that updates the control parameters without detailed system models (which are typically difficult to obtain in a service delivery environment).

We first discuss the workload management problem in IT service delivery and present the architecture of dynamic workload management. Next, we describe the design of the feedback control mechanism as well as uncertainty-based learning and adaption. Finally, we demonstrate the effectiveness of the presented approach in an IT incident management example that is designed based on a large service delivery environment handling more than ten thousand service requests over a period of six months.

2.1 Service Delivery Workload Management

A service delivery environment is generally organized as a set of *service delivery units*, each servicing a subset of customers with similar needs and consisting of a subset of service agents with similar skills. Assigning groups of customers to such service delivery units allows service agents to develop familiarity with customers' IT infrastructure and service requests. This helps to improve quality of service for the service customers and achieve cost efficiencies for the service provider.

One challenge faced by the service delivery provider is to determine the optimal size of a service delivery unit. An oversized service delivery unit results in low staff utilization, while an undersized service delivery unit may encounter difficulty achieving the service level targets. Furthermore, the customer workload varies over time, which adds the complexity for rightsizing. Although some workload variations may be predictable (for example, the banking industry has fewer service requests at

the end of the year when the financial activity slows down), other variations are driven by random factors. The changes in customer behaviors directly impact the workload received by the service delivery units, and subsequently impact their abilities to meet the service level agreements.

Thus, the goal of automated service delivery workload management is to be able to automatically adjust the service delivery unit team size in response to the customer workload changes. Such an adjustment is complicated by the fact that each service delivery unit supports multiple customers with different service level objectives. Furthermore, we need to consider the relationship among the service delivery units when making the adjustment. Because we will not be able to get additional service agents at any given time, the addition of service agents to one service delivery unit necessarily means the reduction of service agents from one or more other service delivery units.

We start from describing the architecture of the dynamic workload management system, as shown in Fig. 2. The feedback controller dynamically assigns or reassigns the service agents to the service delivery units, which handle the service requests from various customers with customer specific service level targets. Service level metrics are periodically measured from the service delivery units, compared to the service level targets, and used by the feedback controller to adjust service agent assignment in response to changes in customer workload.

There are three key modules of the dynamic workload management scheme: service objective alignment, feedback controller, and adaptive design. The goal of *service objective alignment* is to align the high level service level targets from the customers with the low level service level metrics measured from the service delivery environment. This helps to create the real time service operation targets for the service delivery units based on the customers' business objectives regarding service level attainment (or average request resolution time, if specified by the customers), rather than internal operational metrics such as agent utilization or backlog length. The *feedback controller* operates to achieve the service operation target by dynamically adjusting the assignment of service agents to service delivery units. The *adaptive design* module takes the operational data regarding service agent
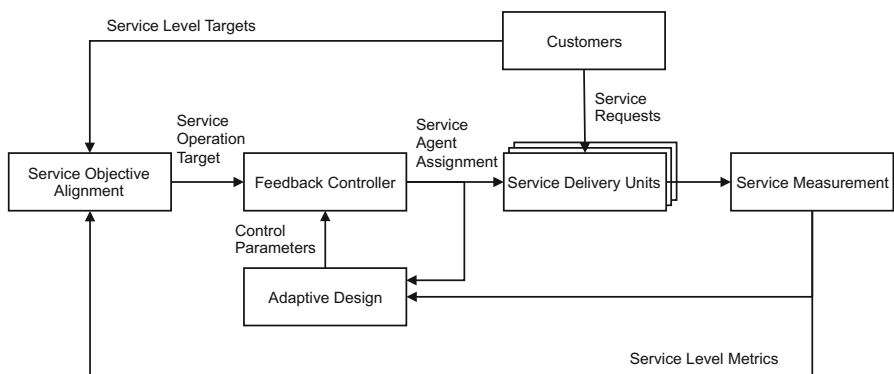

Fig. 2 Architecture of dynamic workload management

assignment and service level metrics as input, and outputs the control parameters for the feedback controller to provide appropriate control performance (i.e., quick response to workload changes while not overreacting to system noise).

2.2 Uncertainty-Based Adaptive Control

Following the architecture discussion above, we describe the design of the dynamic workload management system in detail. This includes (1) service objective alignment to decompose the control strategy for enabling single-input single-output control, (2) feedback control rule construction to adjust service agent assignment based upon feedback from the service level metrics, and (3) adaptive controller design to configure control parameters with minimal modeling effort.

2.2.1 Service Objective Alignment

Feedback controllers implement goal-driven algorithms to track a reference signal and adjust the control input with the objective of minimizing the error between the measured metric and the target reference. In the context of service delivery workload management, the objective is to meet the service level targets from all customers. Fundamentally, this is an optimization problem to minimize the likelihood of missing the service level targets for any customer. Leveraging the "fairness" concept from load balancing, we define the management objective to meet the service level targets equally well (i.e., to keep the same safety margin between the targets and metrics) in order to increase the service resiliency of unknown workload changes. This fairness objective can be further augmented through weighting to consider customer differences (e.g., different service level infringement penalties).

In the following discussion, we describe how feedback controllers can be used to achieve fairness among a set of service level targets from multiple customers. The key method is to decompose the combined targets so that they can be fulfilled by multiple single-input single-output controllers.

Consider a service delivery environment that services M customers from N service delivery units. For each service delivery unit $i, i = 1, \ldots, N$, we define a service operation performance function

$$f_i(SL_i(k), SL_i^*) = \sum_{m=1}^{M} w_m \left(SL_{i,m}(k) - SL_{i,m}^* \right) \tag{1}$$

where $SL_{i,m}^*$ denotes the service level target of service delivery unit i with respect to customer m, $SL_{i,m}(k)$ denotes the measured service level metrics at time interval k, and w_m indicates the weighting factor for customer m. The service level targets can be in the format of the average service request completion time or in the format of the percentage of the service requests that can complete their service within a predefined time interval.

We design the controller to achieve "fairness" among all service delivery units, that is,

$$f_i(SL_i(k), SL_i^*) = f_j(SL_j(k), SL_j^*) \tag{2}$$

for all pairs of service delivery units i and j. This fairness objective can be achieved by constructing a set of single input single output controllers, one for each service delivery unit, with the common reference signal defined as

$$\frac{1}{N} \sum_{j=1}^{N} f_j(SL_j(k), SL_j^*) \tag{3}$$

and the control error for controller i as

$$e_i(k) = f_i(SL_i(k), SL_i^*) - \frac{1}{N} \sum_{j=1}^{N} f_j(SL_j(k), SL_j^*) \tag{4}$$

Thus, the objective of the feedback controller is to drive the control error $e_i(k)$ to zero using the following feedback control law

$$\Delta u_i(k) = u(k+1) - u(k) = K_i(e_i(k)) \tag{5}$$

where $u_i(k)$ denotes the control input (the number of service agents in service delivery unit i). Note that the control law in Eq. (5) looks similar to the integral control law that is used in proportional-integral-derivative (PID) controllers [3]. Although both forms aim to eliminate steady-state errors, the difference is that $K_i(\cdot)$ represents an integral function instead of an integral gain, as in the PID controller. This increases the design flexibility, as we will discuss in the following sections, so that we can design the controller directly based on system behaviors and uncertainties. Such a design is generally simpler than building the difference equation models through system identification, a design that is more involved and has a higher data quality requirement (which is typically lacking in service delivery).

2.2.2 Feedback Control Rule

Although a rich set of controllers has been studied in control literature, the majority are model-based and require extensive modeling and model-based adaptation [4, 5]. Instead, we leverage a simple yet effective control rule based on the Bang-Bang logic. A Bang-Bang (on–off) controller is frequently used in optimal control where the control input is restricted between an upper bound and a lower bound, and an optimal solution is to switch between the control bounds [6]. We apply the Bang-Bang controller due to the coarse granularity of service agents (i.e., they can only be moved in increments of full individuals) and the restricted size of the service delivery units. This makes an on–off controller more applicable, compared to other continuous or discrete control laws.

Figure 3 depicts the operation of the Bang-Bang control law for controller i. The x-axis denotes the control error $e_i(k)$ and the y-axis denotes the change in the control input $\Delta u_i(k)$. A dead zone is defined so that no control input adjustment is given if the control error is between $-d_i$ and d_i. If the control error exceeds the dead zone,

Fig. 3 An illustrative of the
Bang-Bang control law
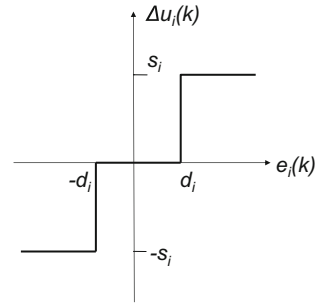
the control input is adjusted by step size s_i. In the case of a positive error (that is, service delivery unit i performs worse than average and performs worse than all service delivery units) the Bang-Bang control law will increase the control input (i.e., add service agents) to reduce the deviation. When control input dependency exists (e.g., the number of service agents assigned to each delivery unit must be positive and the total number of service agents assigned to all delivery units must equal the total number of service agents available), a projection algorithm can be applied.

2.2.3 Adaptive Design

The performance and effectiveness of the above Bang-Bang controller is determined by the careful selection of three control parameters: control interval (T), deadzone size (d_i), and step size (s_i). While Bang-Bang control is widely used in optimal control with rigorous analysis [6], considering the unique complexity and nature of IT service management, we devise a simple and yet effective adaptive design approach that does not rely on detailed modeling and can react to system uncertainties commonly encountered in service delivery.

In a services delivery environment, sources of system uncertainty include the following: (1) *System Randomness*: Service requests are generated at random times and with different levels of complexity. Furthermore, much of the measured data in a service delivery environment depends upon accurate recordings from service agents. Experience has shown that this data does not always reflect real events with perfect precision, due to the discrete (and unpredictable) nature of human beings; (2) *Transient Dynamics*: There are often lags between the time when the events occur and the time when the metrics are measured, which introduces uncertainty as to the true state of the system during the transient phase. For example, after the service agent moves to a different service delivery unit, it will take some time for the agent to settle down, so that a performance lag will be observed before the service level metric improvement is actually realized and reported; (3) *Workload Variation*: Both system configurations (e.g., assignment of customer accounts to the service delivery units) and workload behaviors (e.g., service request arrival rate per customer, service rate at which service agents handle service requests) can vary over

time. Thus, robustness and adaptability are desired for a feedback controller that operates in a real environment.

By using the Bang-Bang control logic and the three control parameters defined above, we design the adaptive feedback controller to accommodate these sources of system uncertainty.

Control Interval We determine the control interval from the perspective of understanding and managing system randomness. Intuitively, if data variability is high, a larger control (sample) interval is required to ensure meaningful service measurement and control. According to the Central Limit Theorem, the distribution of the sample average of random variables approaches the normal distribution with a mean equal to that of the parent distribution and variance equal to that of the parent divided by the sample size (N), irrespective of the parent distribution.

For an initial control interval T_0, we measure the service level metrics $SL_i(k)$ and calculate the mean μ_{SL_i} and the standard deviation σ_{SL_i}. Given the desired noise ratio $r = \frac{\sigma_{SL_i}}{\mu_{SL_i}}$ from the control designer, we can compute the control interval (T) as follows:

$$\sigma^*_{SL_i} = r\mu_{SL_i} = \frac{\sigma_{SL_i}}{\sqrt{N_i}} \tag{6}$$

$$T = T_0 \max N_i \tag{7}$$

Based on experience, we set $r = 0.1$ to balance the feedback controller between sensitivity to system randomness and ability to adapt to workload changes.

Dead Zone Size The dead zone is used to increase controller robustness to system randomness and workload variation. Since the impact of system randomness cannot be entirely eliminated using control interval selection, the dead zone is introduced to avoid control oscillation, especially around the optimal steady state when the control error appears small. A dead zone is also valuable when the control input has a coarse granularity. For example, service agents can only be reassigned in increments of full individuals, even if the theoretical optimal value indicates a fractional adjustment. We design the dead zone size as follows:

$$d_i = l\mu_{SL_i} \tag{8}$$

where l is the threshold limit that makes the dead zone size proportional to the average of the service level metric. Typically, we choose $l = 2r$, with the objective that no control action should be reacting to system randomness. Generally, a larger threshold limit can reduce oscillation but may also lead to a larger steady state error.

Step Size The step size is related to the speed of the controller convergence. A larger step size results in a faster controller response regarding workload variation, but may cause the controller to oscillate around the optimal point with control error bouncing around the dead zone. Conversely, a smaller step size leads to a longer convergence time. From our experience, we choose an initial step size s equal to 5% of the control range (the number of service agents in the service delivery unit).

If the step size is too large and causes oscillation around the dead zone, we introduce an oscillation-induced adaptation algorithm to resize the dead zone, as

follows: (1) Observe the control input history and record the sign of control input change; (2) If an oscillation pattern is detected (e.g., the number of increases is equal to the number of decreases, e.g., $1, -1, 1, -1$), increase the dead zone size by 20%; (iii) If a chasing pattern is detected (e.g., $1, 1, 1, 1,$ or $-1, -1, -1, -1$), decrease the dead zone size by 20%.

2.3 Model Evaluation

Our evaluation is based on the incident management data collected from a large service delivery environment over a period of six months, including more than ten thousand service requests. Each incident service request includes details about the incident arrival time, service time, and completion time. A separate data source provides the service level agreement information with the target service response time and target service level attainment.

We analyze the service requests in order to characterize the workload. (For the purposes of this evaluation, some workload parameters have been modified to ensure sensitive business data are not revealed.) Our objective is to build a service testbed (simulator) based on a set of queueing models (specifically, M/M/m models [7]), that are calibrated using the collected service request data. We use the model to represent the dynamic behavior of the service delivery environment and examine how the presented controller can be used for dynamic service agent assignment.

Table 1 lists the workload parameters for our evaluation. The workload arrives from four customers. The inter-arrival times of the service requests follow exponential distributions, as approximated from the collected data. We evaluated goodness-of-fit using the Kolmogorov–Smirnov test where the test statistic, the least upper bound for the cumulative distribution function, is 0.17 for the exponential distribution, compared to, for example, 0.28 for the normal distribution, and 0.70 for the lognormal distribution. Table 1 also summarizes the workload distribution across the two service delivery units. In addition, the average service time per service request is 62.2 min in service delivery unit 1 and 123.9 min in service delivery unit 2. These service times follow exponential distributions as well. The target service response time is 8 hours for all service requests, and the service level attainment target is 95% for all customers in each of the service delivery units. This is a simplification used in this queueing model; in a separate effort, we are building a discrete event simulation model for better representation. There is a total of 20 service agents in the two service delivery units. The control decision to be made in

Table 1 Workload parameters from an IT incident management example

|	Customer 1	Customer 2	Customer 3	Customer 4
Inter-arrival time (min)	14.2	45.4	177.1	20.1
Service Delivery Unit 1 (%)	92.5	83.8	64.4	82.1
Service Delivery Unit 2 (%)	7.5	16.2	35.6	17.9

each control interval is the allocation of service agents to each of the service delivery units, with the objective of fairly meeting the service level attainment targets.

Figure 4 displays the performance of the presented feedback control mechanism. The initial response is shown in Fig. 4a. The top and middle plots show how the service agents and the service attainment levels converge. The bottom plot displays the control error (the solid line) and the dead zone (the two dashed lines). As the control error enters the dead zone, no more service agent adjustments are performed. Figure 4b displays the controller performance when the workload changes at interval 10. The workload distribution between service delivery units 1 and 2 changes to 62.5 and 37.5%, respectively, for customer 1. This could occur, for example, if customer service requests follow seasonal patterns, or if the service delivery provider changes the assignment of customers (or types of customer service requests) to service delivery units. The controller begins with the optimal setting from Fig. 4a. When the workload changes, service attainment increases for service delivery unit 1 since fewer service requests arrive, prompting the controller to shift service agents to service delivery unit 2. After measuring the service level metrics, the controller modifies the service agent allocation to nine service agents in service delivery unit 1 and eleven service agents in service delivery unit 2.

Overall, the proposed control mechanism successfully balances service level attainment across service delivery units, both when the workload pattern is stationary over time and when the workload dynamically changes. Further, appropriate choice of step size combined with dead zone definition prevents the controller both from oscillating and from responding to system noise.

2.4 Lessons Learned

The design of the above service delivery workload management system follows the monitoring, analysis, planning, and execution ("MAPE") model from autonomic
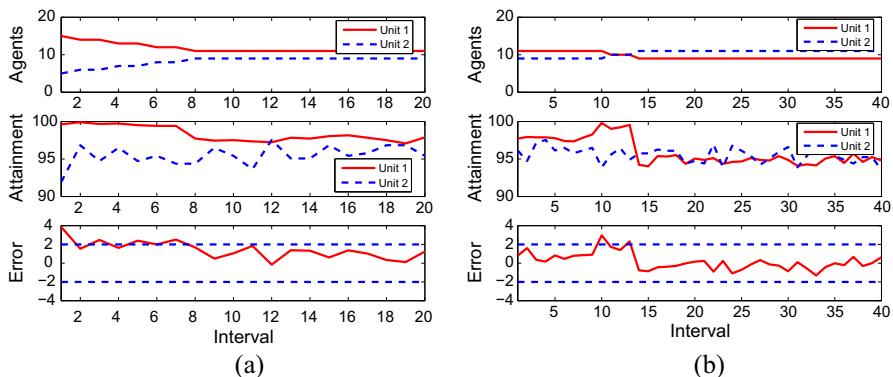
Fig. 4 Control performance of the feedback controller. The x axes indicate the control (sample) interval, where each interval is 10 days. The y axes indicate the number of agents in the top plot, the service level attainment percentage in the middle plot, and the error of the service level attainment percentage in the bottom plot. **a** Unknown workload, **b** workload change

computing. It demonstrates the possibility of building autonomic systems for IT service management using a similar feedback control structure for IT infrastructure management. However, beyond the similarity at the principle level, a properly designed autonomic service management system needs to consider the significant uncertainty in a service delivery environment and respond to it accordingly (e.g., through an uncertainty-based adaptive control design as presented in this section). Furthermore, building autonomic systems for IT service management needs to consider the real complexity of IT services and service agents, if it is to be used for complex decision making, which will be illustrated in the next section.

3 Simulation-Based Optimization for Workforce Management

In this section we describe an automated approach for workforce management [8]. This supports the *Capacity Management* process within ITIL's *Service Design* lifecycle. The presented approach employs the simulation optimization methodology to provide recommended staffing levels that balance the conflicting needs associated with dynamic customer workload, strict service level constraints, and service personnel with diverse skill sets. Compared to the autonomic workload management approach discussed in the previous section, the presented approach in this section differentiates itself by focusing on off-line service delivery team composition to determine both the agent number and agent skills required in each service delivery unit to respond to different types of service requests, as well as the shift schedules that the service agents must follow. Such decision making involves much more complexity in the decision variables and the optimization constraints, as compared to the real time workload management in regard to reassigning the agents from one service delivery unit to another.

We start the section by describing the simulation optimization model in detail. Afterwards, we demonstrate the effectiveness of the presented approach in an IT service delivery environment. Indeed, the extension of the presented simulation optimization framework has been implemented and deployed at a large services delivery provider with worldwide delivery locations and global customers.

3.1 Simulation Optimization Model

We define the staffing level as the number of service agents within the service delivery unit, the shift schedule of the service delivery unit with respect to which agent works on what time, and the required agent skill profile in order to resolve the customer service requests. The goal of the workforce management is to find a minimal staffing level with respect to the agent cost and, at the same time, be able to resolve the customer service requests within the service agreements. That is, the objective of the simulation optimization model is to minimize the total staffing related variable cost while considering the contractual service level constraints, the skills required to respond to different types of service requests, and the shift schedules that the service agents need to follow.

There are a number of complexities in a service delivery system that need to be considered properly. First, the number of classes of requests arriving to the system is large, since the requests are differentiated by the different attributes, and the request arrival rates are non-stationary, varying over the hours of the day and days of the week. Second, the agents from different service delivery units have different breadth and depth of skills and are working on different shift hours. Third, the service target time can be measured either against calendar hours or business hours; in the latter case, a business calendar is required. Although complicated first principle models (e.g., [9]) or queueing models (e.g., [10, 11]) can be built to approximate complex systems, they lack the level of detail and accuracy to address the above complexities encountered in the service delivery systems. It is due to these modeling complexities as well as the inherent stochastic nature of the problem that we choose a simulation-based modeling and optimization framework to determine optimal staffing levels.

We first define the basic elements and assumptions in our model. Next, we specify the objective function and the constraints, and introduce the simulation model that we use to characterize the delivery system operation. Finally, we discuss the solution techniques. The notation used in this paper is summarized in Table 2.

Table 2 Notation for workforce management

$i = 1, \ldots, I$	Set of customers
$j = 1, \ldots, J$	Set of shift schedules
$k = 1, \ldots, K$	Set of service delivery units
$c = 1, \ldots, C$	Set of request complexities
$p = 1, \ldots, P$	Set of request priorities
$t = 1, \ldots, T$	Set of intervals in the workload horizon
$l = 1, \ldots, L$	Set of intervals in the shift schedule horizon
$r = 1, \ldots, R$	Set of staffing equality constraints
x_{jk}	Number of agents assigned to shift schedule j in service delivery unit k
$\bar{x}_k$	Upper bound on the number of agents assigned to service delivery unit k
$\underline{x}_k$	Lower bound on the number of agents assigned to service delivery unit k
$\bar{x}_j$	Upper bound on the number of agents assigned to shift schedule j
$\underline{x}_j$	Lower bound on the number of agents assigned to shift schedule j
c_k	Cost of agent in service delivery unit k
b_r	Staffing equality constant for the r-th constraint
a_{kr}	Staffing equality parameter for the r-th constraint at delivery unit k
v_{ipc}^t	Average volume of workload in period t for customer i with priority p and complexity c
s_{ipc}	Average service time for customer i with priority p and complexity c
y_{jl}	$\begin{cases} 1 & \text{if shift } j \text{ is staffed in period } l \\ 0 & \text{otherwise} \end{cases}$
α_{ip}	SLA attainment target for customer i of priority p
w_{ip}	SLA target time for customer i of priority p
m_{ip}	SLA measurement period for customer i of priority p

3.1.1 Definitions and Assumptions

Let $i = 1, \ldots, I$ denote the set of customers. Let $k = 1, \ldots, K$ denote the set of service delivery units in which the service agents are organized. All agents within the same service delivery unit have the same depth and breadth of skills (e.g., technical skills, customer environment familiarity) and can respond to the arrived service request equally. Let $j = 1, \ldots, J$ be the set of allowable shift schedules to which agents could potentially be assigned. These shift schedules are ensured to provide adequate customer service coverage (e.g., 24 by 7) and to satisfy the regulatory requirements (e.g., total number of hours worked per week or consecutive working hours per day).

Arriving service requests are classified by customer i, as well as by complexity $c = 1, \ldots, C$ based upon the skills required to respond to these requests. The request complexity can be defined to have a one to one mapping to the service delivery unit defined above; alternatively, a mapping table can be used for more complicated mapping. The service requests are further distinguished by their priorities $p = 1, \ldots, P$. We use priority to characterize the severity and urgency of the service requests. The priority level specifies the order in which the arrived request will be processed by the service agent.

Note that, given the complexity of the service delivery system, a sizable number of parameters are used in the model formulation. However, we design the model so that all required parameters can be commonly measured.

3.1.2 Objective Function and Constraints

The workforce management problem can be formulated in two different ways. The first formulation includes both the staffing cost and the SLA violation cost in the objective function and the goal is to minimize the sum of these two costs. This allows the service delivery provider to trade between the cost of hiring additional staff and the cost of incurring additional service quality penalties due to lack of sufficient staffing. The second formulation only includes the staffing cost in the objective function but models the service level agreement as a constraint that must be satisfied.

Although mathematically feasible, the first approach is less likely to be adopted by the service provider from the business point of view. Quality of service and service level attainment are metrics that both service delivery providers and customers monitor on a continual basis. Trading off staffing cost against service level attainment does not consider the cost of good will and may very likely lead to customer dissatisfaction. We therefore adopt the latter formulation and state the optimization problem as follows:

$$\min \sum_{j=1}^{J} \sum_{k=1}^{K} c_k x_{jk} \tag{9}$$

$$s.t. \quad f_{ip}(v^t_{ipc}, s_{ipc}, y_{jl}, x_{jk}, w_{ip}, m_{ip}) \leq \alpha_{ip} \quad (10)$$

$$\underline{x}_k \leq \sum_{j=1}^{J} x_{jk} \leq \overline{x}_k \quad (11)$$

$$\underline{x}_j \leq \sum_{k=1}^{K} x_{jk} \leq \overline{x}_j \quad (12)$$

$$\sum_{j=1}^{J} \sum_{k=1}^{K} a_{kr} x_{jk} = b_r \quad (13)$$

$$x_{jk} \geq 0 \quad (14)$$

Equation (9) defines the staffing cost to be minimized where x_{jk} denotes the number of service agents organized in the k-th service delivery unit and assigned to the j-th shift. We also define the cost variables c_k as the unit cost per service agent within the k-th service delivery unit (noting that the delivery units are organized based on the depth and breadth of the skills and highly skilled agents generally demand higher cost). The total staffing cost is summed over agents from all delivery units at all shifts.

We consider the following two types of constraints: service level constraints and staffing coverage constraints. *Service level constraints*, as defined in Eq. (10), represent the service level objectives that must be satisfied. In a service delivery environment, the service level objectives typically take on a form such as "no more than 5% of priority 1 incidents reported each month can be resolved in more than 2 calendar hours." We use α_{ip} to denote the attainment target associated with the class of service requests from customer i with priority p, w_{ip} to define the SLA target time, and m_{ip} to represent the measurement period. Due to the complexity of service delivery operation, we use discrete event simulation $f(\cdot)$ to compute the service attainment level (or, more precisely, the service violation level) from a number of factors [12].

We define v^t_{ipc} as the volume of service requests arriving from customer i with priority p and complexity c during the time period t. The variability in the arriving workload is stochastic in nature over short periods of time but exhibits a repeating weekly pattern. We model the arrival of workload as a non-homogeneous Poisson process. That is, we assume the arrival rate follows a stationary Poisson arrival process within each of one hour time periods for $t = 1, \ldots, T$ ($T = 168$) hours of the week.

We note that some other temporal patterns also exhibit in the data (e.g., due to quarterly or end of year changes). However, we choose not to consider these for the following reasons: First, due to the dynamic nature of the service delivery environment, it is typically difficult to obtain long periods of historical data that are stable enough to derive the seasonal patterns. Second, although seasonal workload patterns do exist, the service delivery units are typically able to use alternative

means (other than changes in staffing) to manage the end of quarter / end of month peaks in workload. For example, certain non-demanding workload types (e.g., documentation update, knowledge transfer) can be scheduled during non-peak periods. Third, scheduled overtime can be used to meet some excess demand during periods of high demand, though overtime is not relied upon too extensively due to either regulatory constraints or the already long shifts worked by agents. Finally, the agents' vacations are typically scheduled in consideration of these known seasonal workload patterns. Thus, for example, more agents take vacation or holiday in August or December when the volume of work decreases for many of the service delivery units, and coincidently at the same time when most of the customers are on vacation or holiday too. In summary, the weekly workload pattern has been shown to be sufficient to satisfy our practical needs in the context of staffing optimization.

In addition to the workload volume, the simulation model also takes the request service time s_{ipc} for customer i with priority p and complexity c. Similar to the findings of [13] and supported by the theoretical work of [14], based on data collected from many service delivery units we find that the distribution of the service times is well modeled by a lognormal distribution. Finally, we use y_{jl} to denote the shift working hours for schedule j at period l; $y_{jl} = 1$ if shift schedule j is staffed in period l and 0 otherwise (where $l = 1, \ldots, L$ and L defines the periodicity after which the schedule repeats itself).

Staffing coverage constraints, as defined in Eqs. (11–13), represent the restrictions on the staffing assignment. Equation (11) places restrictions on the number of agents within each service delivery unit. For example, there may be a limited number of high skilled agents available so that the maximum size of the "high skill" service delivery unit is limited. We use $\bar{x}_k$ to state the upper bound of the number of agents in delivery unit k; similarly, a lower bound $\underline{x}_k$ is defined.

In some cases, there are constraints on the number of agents who must work in a given shift. These may be "physical constraints" due to the configuration of the delivery environment. For example, agents may be required to monitor consoles and due to the physical layout of the delivery environment there is a minimum number of agents who must be available to monitor the consoles irrespective of the workload. In other examples, customer contracts may specify a minimum number of agents who must be available during certain shifts. Equation (12) is used to capture these constraints where $\bar{x}_j$ and $\underline{x}_j$ denote the upper bound and lower bound of the number of agents in shift j.

Finally, there are cases where the number of agents from certain delivery units is fixed or a mix of them is fixed, these are captured in Eq. (13) as equality constraints where b_r defines the equality constant for the r-th constraint and a_{kr} defines the corresponding coefficient for the k-th delivery unit.

3.1.3 Solution Approaches

The workforce management problem described in Eqs. (9–14) defines a feasible region (or a search space) to find the optimal staffing levels. Considering the implementation practicability, we adopt the intelligent search procedures inherent in

the *Scatter Search* combined with *Tabu Search* metaheuristics (see, e.g., [15, 16]) to find the optimal solution. This is because of the complexity of service delivery and the fact that we rely on the simulation model to accurately represent the delivery system dynamics. As such, we will not be able to use analytical techniques to determine the feasibility of the evaluated solutions.

Scatter search originated from strategies for creating composite decision rules and surrogate constraints. It generates a reference set of trial solutions, and improves the solutions by joining solutions based on generalized path constructions in Euclidean space. To enhance the convergence of scatter search, tabu search is applied to recall the performance of proposed solutions that have been evaluated and guide the search process. It ensures that solutions that have already been evaluated will not be reevaluated, guides intensification or diversification of the search, and leads the search away from a locally optimal solution.

The optimizer based on the above approaches makes no mathematical assumptions for the functions within the feasible region. This makes it particularly useful for solving the optimization problem with embedded simulation models, where the solutions suggested by the optimizer are evaluated by the simulation model for performance. Afterwards, the output of the "performance test" is returned to the optimizer and forms a continual feedback loop.

The execution time of the optimization model depends on two factors: the simulation time of each iteration and the number of iterations needed to converge to the optimal state. For the workforce management problem, the simulation time is mainly affected by the workload volume for each customer/priority group. This is because the simulation needs to run long enough to make the SLA measurement statistically stable. The number of iterations is mainly determined by the number of service agents, the number of service delivery units, and the number of shift schedules. The combination of them defines the optimization space. In our experience, the convergence time is normally on the order of hours. Since workforce management is typically part of strategic planning that occurs at a much slower time scale (usually on the order of months), the convergence time is of limited concern.

3.2 Model Evaluation

We provide an experimental evaluation to illustrate how the presented simulation optimization model can be be used to produce staffing decisions in a services delivery organization. Our implementation is built on top of the AnyLogic simulation software, which supports agent-based, system dynamics, and discrete event simulation methodologies in an visual development environment [17, 18]. It also provides several optimization packages (including Scatter Search and Tabu Search that we are using) for Monte Carlo simulations. Using AnyLogic, we specify and code appropriate constraints and performance criteria to yield a reasonable feasible region and to quickly converge to an implementable solution. The developed model is exported as a standalone Java application for users to run models independently. Note that the data has been altered to preserve data privacy and simplified for the illustration purpose, though the nature of day-to-day service operations has been maintained.

We consider three types of service requests with different priorities: console alerts, problem tickets, and ad hoc requests. In Table 3, we provide basic statistics for these requests. The requests differ in their arrival rate distributions, service time distributions, and service level objectives. The console alerts have the highest weekly volume and most stringent service attainment target, and are serviced with the highest priority. The problem tickets have a low volume but longer service time and have the medium priority. The ad hoc requests have the highest average and variability in the service time, but are serviced with the lowest priority.

We model the service time using a lognormal distribution. Initially, we model the arrival rate using a stationary Poisson arrival process; afterwards, we use the non-homogeneous Poisson process to capture the varying arrivals for each hour of the week. We use this as an example to demonstrate the need of the simulation model (compared to analytical models) since it can conveniently consider multiple complex factors such as non-homogeneous Poisson process and agent shift schedules.

Figure 5 displays the convergence path of the simulation optimization model. The x-axis indicates the number of iterations (i.e., the number of solutions that have been created and evaluated) and the y-axis indicates the total number of agents in each solution. The oscillating line in the background shows all candidate solutions from scatter search and tabu search (including both feasible and infeasible solutions), the line in the middle shows the evolution of the best feasible solutions (i.e., the ones that meet all constraints especially the service level constraints), and the line at the bottom shows the minimum (but infeasible) solutions that have been evaluated. The model starts from a random configuration and finds the first feasible solution at the second iteration (it happens in this example, but generally the first feasible solution may be found much later). The model converges after 102 iterations, during which 53 solutions have been evaluated and five of them are feasible solutions. A solution only needs to be evaluated if the total number of agents from this solution is smaller than that from the current best feasible solution (i.e., below the line in the middle), even though the ones above the line also need to be generated in order to construct the generalized path.

Figure 6 shows the per-shift staffing levels for the five feasible solutions found during this optimization process. Each cluster of bars represents the number of agents recommended for each shift, and the fifth column indicates the total number of agents. The last cluster shows the optimal staffing configuration with 32 agents distributed evenly across four shifts (i.e., eight agents per shift). The balanced shift assignments reflect the stationary arrival of workload that we artificially assumed in

| Table 3 Scenario of service requests	Alerts	Problems	Ad hoc
Average weekly volume	6784	403	940
Average service time (min)	5.47	12.40	20.30
Stdev service time	3.46	11.77	23.48
Target response time (min)	10	30	60
Service attainment target (%)	99	95	95
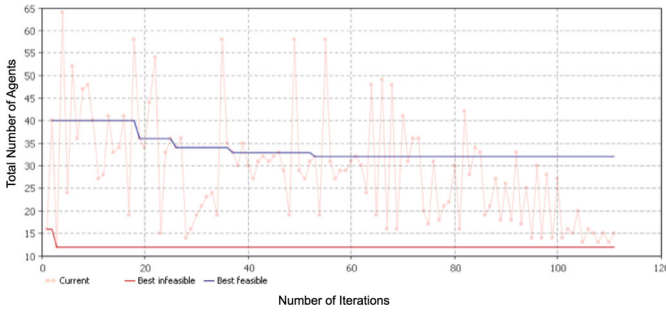
Fig. 5 Convergence process of staffing configuration optimization
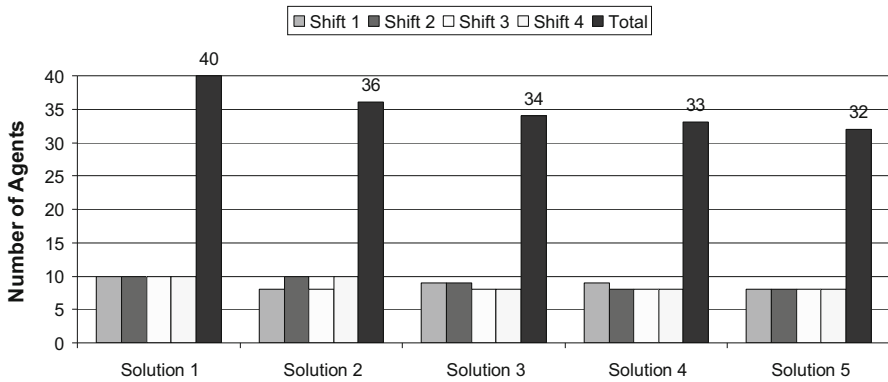
Fig. 6 Number of agents recommended in each feasible solution

the beginning of this section. Further, the agents are, on average, fully utilized over all of their available working hours.

Next, we use the non-homogeneous Poisson process to capture the varying arrivals for each hour of the week. Figure 7 illustrates the volume of arrivals per hour of the week. The x-axis indicates the hours in the week where hour 1 is the hour between Sunday midnight and 1 am Monday morning. The y-axis indicates the total volume of alert requests for the corresponding hour of the week. We run the optimization model similarly to the last scenario. However, the new optimal solution requires 64 service agents, twice of that under the stationary arrival scenario. Examination of the staff utilization reveals low average utilization of 52.2% across all shifts. Apparently, the peaks and valleys as observed in Fig. 7, even though not significant, do have a drastic impact on the staffing level. This is because of the short target response times and strict service attainment targets, so that many more agents are required to ensure that the service level agreements can be met at the peak of the week.
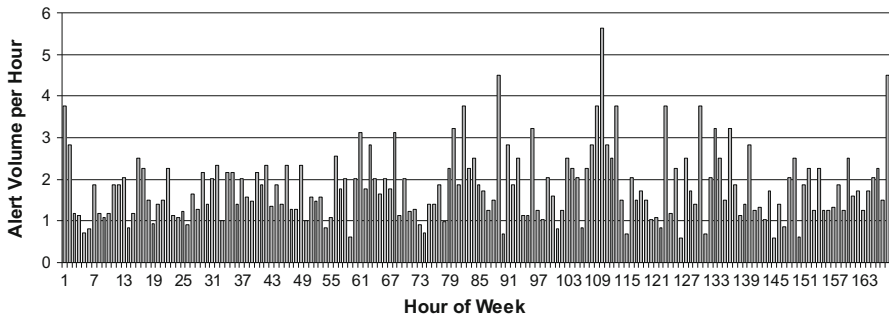
Fig. 7 Volume of alerts per hour of the week

3.3 Lessons Learned

As another example of building autonomic systems for IT service management, the simulation optimization approach presented in this section determines minimum staffing requirements while meeting contractual service quality commitments. The approach utilizes discrete event simulation as the analysis module to capture the complex relationships within a service delivery system, and scatter search combined with tabu search as the planning module to find the optimal staffing level in a complex global services delivery environment. Given the complexity in real service delivery environments, a more complicated autonomic management system is needed in contrast to the rather simple approach discussed in the previous section. However, the presented simulation optimization model is designed with a reasonable set of operational and demographic data, which lends its applicability in integrating with actual service delivery decision making.

4 Recommender System for Event Management

In this section we describe an autonomic approach for resolution recommendation [19]. This supports the *Event Management* process within ITIL's *Service Operation* lifecycle. The presented approach employs the topic-level feature extraction technique over the event and resolution information from historical service requests to recommend appropriate resolutions for incoming events. Compared to existing approaches in IT service management and recommender systems, the presented approach differentiates itself by (1) taking into account the false tickets often generated by monitoring systems for intermittent conditions, (2) designing the topic-level features to incorporate the resolution information into the similarity measurement, and (3) using metric learning to learn a more effective similarity measure.

We first overview the event management problem and the architecture of resolution recommendation. Next, we discuss in detail our four approaches to improvement the recommendation quality. Finally, we demonstrate the

effectiveness and efficiency of the discussed methods through empirical evaluations on three service data sets.

4.1 Service Monitoring and Event Management

Problem detection is the first step in the problem detection, determination, and resolution workflow, which is usually conducted by the monitoring software to emit events when anomalous behaviors are detected. Towards fully automated service monitoring and event management, an automated response system provides an effective and reliable means of ensuring that anomalous behaviors or degradation of the vital signs of IT systems or workload are flagged, automatically analyzed for being non-redundant and genuine, and finally automatically resolved or sent to the system administrators with resolution recommendation.

Each event has several related attributes with values describing the system status at the time the event is generated. For example, a CPU-related event usually contains the CPU utilization and paging utilization information, and a capacity-related event usually contains the disk name and the size of disk used/free space. An event also contains a textual description of the service interruption, called the event summary. When subsequent actions are needed to resolve the incident and bring the service back to normal, the event summary will become the problem description of an incident ticket (a.k.a., service request) to be acted upon by the service agent. A problem resolution is also stored in the incident ticket as a textual description of the steps taken to resolve the problem.

In this section, we describe a method that finds a resolution for an event by making use of similarities between the events and previous resolutions of monitoring tickets. From analyzing historical monitoring tickets collected from three customers managed by a large service provider, we observe that there are many repeating resolutions for monitoring tickets within the same customer. Naturally, if the events are similar, their respective tickets will most likely share the same resolution. Therefore, it is possible for us to design an automated system that recommends the proper resolution for an incoming ticket based on its corresponding event information and the history of the resolved tickets.

Figure 8 shows the architecture of resolution recommendation. We start from the traditional k-nearest neighbors (KNN) algorithm [20]. It uses the attribute level features to provide resolution recommendations for incoming tickets in event management (Sect. 4.2.1). We use four approaches to extend the basic KNN in order to improvement the recommendation quality. In our first approach (LDABaselineKNN), we improve the traditional KNN for event similarity measure by using the weighted KNN (WKNN) and introducing the topic level features through the Latent Dirichlet Allocation (LDA) method (Sect. 4.2.2). The second approach is used for handling the false tickets by using the Division method to identify the false tickets for separate handling (Sect. 4.2.3) and the Probabilistic Fusion method to incorporate an additional penalty for minimizing the misleading resolutions (Sect. 4.2.4). Our third approach (CombinedLDAKNN) targets both the event and the corresponding ticket resolution information with the top level features (Sect. 4.2.5). Finally, our fourth approach introduces metric Learning with a
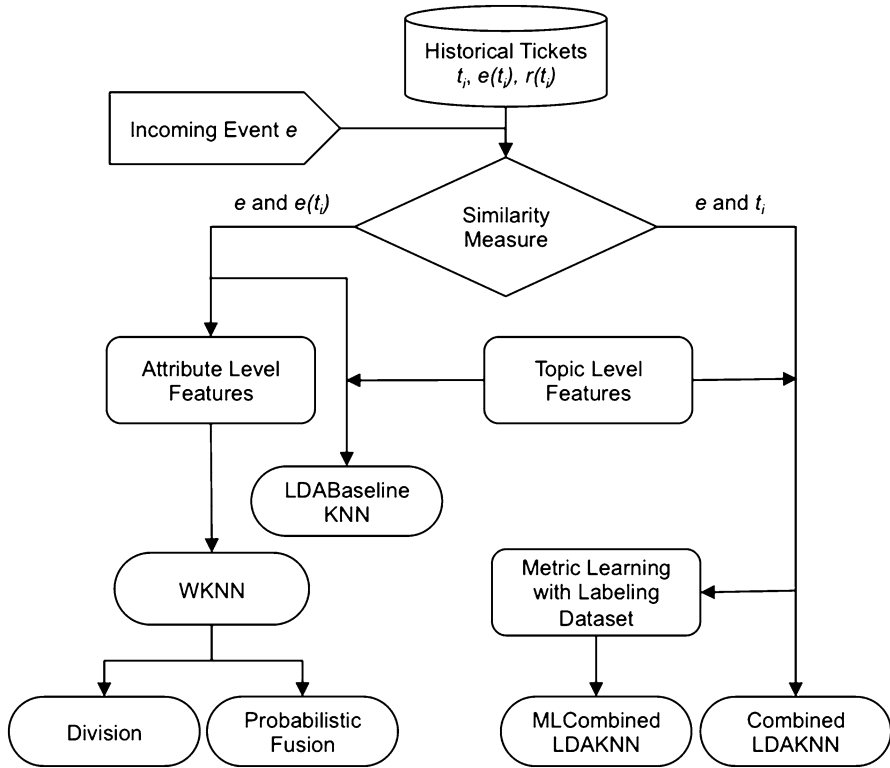
Fig. 8 Architecture of resolution recommendation

labeling dataset when the ticket resolution category information is available (Sect. 4.2.6).

4.2 Resolution Recommendation Using KNN-based Extensions

Given an incoming ticket, the objective of the resolution recommendation is to find k resolutions as close as possible to the true one for some user-specified parameter k from the historical tickets. Each historical ticket has attributes of categorical, numerical and textual types. An incoming ticket is one that has no resolution component. Moreover, tickets are encoded as feature-level attributes in which each ticket is considered as a composition of attributes, and the similarity between tickets is calculated as the sum of the attributes' similarity according to Eq. (15), or topic-level attributes in which each ticket is considered as a probability distribution of topics extracted using approach described later.

4.2.1 Basic KNN-Based Recommendation

The recommendation problem is often related to that of predicting the top k possible resolutions. A straightforward approach is to apply the KNN algorithm, which

searches the K nearest neighbors of the given ticket (K is a predefined parameter), and recommends the top $k \leq K$ representative resolutions among them [21, 22]. The nearest neighbors are indicated by similarities of the associated events of the tickets. In this paper, the representativeness is measured by the number of occurrences in the K neighbors.

Table 4 lists the notations used in this paper. Let $D = \{t_1, \ldots, t_n\}$ be the set of historical monitoring tickets and t_i be the i-th ticket in D, $i = 1, \ldots, n$. Given a monitoring ticket t, the nearest neighbor of t is the ticket t_i that maximizes $sim(e(t), e(t_i))$, $t_i \in D$, where $sim(\cdot, \cdot)$ is a similarity function for events. Each event consists of event attributes with values. Let $A(e)$ denote the set of attributes of event e. The similarity for events is computed as the summation of the similarities for all attributes. There are three types of event attributes: categorical, numeric and textual.

Given an attribute a and two events e_1 and e_2, $a \in A(e_1)$ and $a \in A(e_2)$, the values of a in e_1 and e_2 are denoted by $a(e_1)$ and $a(e_2)$. The similarity of e_1 and e_2 with respect to a is

$$sim_a(e_1, e_2) = \begin{cases} I[a(e_1) = a(e_2)], & \text{if } a \text{ is categorical}, \\ \dfrac{|a(e_1) - a(e_2)|}{max|a(e_i) - a(e_j)|}, & \text{if } a \text{ is numeric}, \\ Jaccard(a(e_1), a(e_2)), & \text{if } a \text{ is textual}, \end{cases}$$

where $I(\cdot)$ is the indicator function returning 1 if the input condition holds and 0, otherwise. Let $max|a(e_i) - a(e_j)|$ be the size of the value range of a. $Jaccard(\cdot, \cdot)$ is the Jaccard index for the *bag of words model* [23], frequently used to compute the similarity of two texts. Its value is the proportion of common words in the two texts. Note that for any type of attribute, inequality $0 \leq sim_a(e_1, e_2) \leq 1$ holds. Then, the similarity for two events e_1 and e_2 is computed as:

Table 4 Notations

Notation	Description
D	Set of historical tickets
$|\cdot|$	Size of a set
t_i	i-th monitoring ticket
$r(t_i)$	Resolution description of t_i
$e(t_i)$	The associated event of ticket t_i
$c(t_i)$	Type of ticket t_i, $c(t_i) = 1$ indicates t_i is a real ticket, $c(t_i) = 0$ indicates t_i is a false ticket
$A(e)$	Set of attributes of event e
$sim(e_1, e_2)$	Similarity of events e_1 and e_2
$sim_a(e_1, e_2)$	Similarity of a values of event e_1 and e_2
K	Number of nearest neighbors in the KNN algorithm
k	Number of recommended resolutions for a ticket, $k \leq K$

$$sim(e_1, e_2) = \frac{\sum_{a \in A(e_1) \cap A(e_2)} sim_a(e_1, e_2)}{|A(e_1) \cup A(e_2)|}. \tag{15}$$

Clearly, $0 \leq sim(e_1, e_2) \leq 1$. To identify the type of attribute a, we only need to scan all appearing values of a. If all values are composed of digits and a dot, a is numeric. If some value of a contains a sentence or phrase, then a is textual. Otherwise, a is categorical.

4.2.2 Division Method

Traditional recommendation algorithms focus on the accuracy of the recommended results. However, in automated service management, false tickets are unavoidable in both the historical and incoming tickets [24].

Since overlooking a real system problem may have serious consequences, we consider incorporation of penalties in the recommendation results. There are two cases meriting a penalty: recommendation of a false ticket's resolution for a real ticket, and recommendation of a real ticket's resolution for a false ticket. The penalty in the first case should be larger since the real ticket is more important. The two cases are analogous to the *false negative* and *false positive* in prediction problems [22], but note that our recommendation target is the ticket resolution, not its type. A false ticket's event may also have a high similarity with that of a real one. The objective of the recommendation algorithm is now maximized accuracy under minimized penalty.

A straightforward solution consists in dividing all historical tickets into two sets comprising the real and false tickets, respectively. Then, it builds a KNN-based recommender for each set, respectively. Another ticket type predictor is created, establishing whether an incoming ticket is real or false, with the appropriate recommender used accordingly. The divide method works as follows: it first uses a type predictor to predict whether the incoming ticket is real or false. If it is real, then it recommends the tickets from the real historic tickets; if it is false, it recommends the tickets from the false historic tickets. The historic tickets are already processed by the system administrator, so their types are known and we do not have to predict them.

The division method is simple, but relies heavily on the precision of the ticket type predictor, which cannot be perfect. If the ticket type prediction is correct, there will be no penalty for any recommendation result. If the ticket type prediction is wrong, every recommended resolution will incur a penalty. For example, if the incoming ticket is real, but the predictor says it is a false ticket, then this method only recommends false tickets. As a result, all the recommendations would incur penalties.

4.2.3 Probabilistic Fusion Method

To overcome the limitation of the division method, we develop a probabilistic fusion method. The framework of the basic KNN-based recommendation is

retained, with the difference that the penalty and probability distribution of the ticket type are incorporated in the similarity function.

Let λ be the penalty for recommending a false ticket's resolution for a real ticket, and $1 - \lambda$ that for recommending a real ticket's resolution for a false one. λ can be specified by the system administrator based on the actual cost of missing a real alert, $0 \le \lambda \le 1$. Larger λ indicates a greater importance of real tickets. The penalty function is

$$\lambda_t(t_i) = \begin{cases} \lambda, & t \text{ is a real ticket}, t_i \text{ is a false ticket} \\ 1 - \lambda, & t \text{ is a false ticket}, t_i \text{ is a real ticket} \\ 0, & \text{otherwise}, \end{cases}$$

where t is the incoming ticket and t_i is the historical one whose resolution is recommended for t. Conversely, an award function can be defined as $f_t(t_i) = 1 - \lambda_t(t_i)$. Since $0 \le \lambda_t(t_i) \le 1$, $0 \le f_t(t_i) \le 1$.

Let $c(\cdot)$ denote the ticket type. $c(t_i) = 1$ indicates t_i is a real ticket; $c(t_i) = 0$ indicates t_i is a false ticket. Since t is an incoming ticket, the value of $c(t)$ is not known. Using a ticket type predictor, we can estimate the distribution of $P[c(t)]$. The idea of this method is to incorporate the expected award in the similarity function. The new similarity function $sim'(\cdot, \cdot)$ is defined as:

$$sim'(e(t), e(t_i)) = E[f_t(t_i)] \cdot sim(e(t), e(t_i)), \tag{16}$$

where $sim(\cdot, \cdot)$ is the original similarity function defined by Eq. (15), and $E[f_t(t_i)]$ is the expected award, $E[f_t(t_i)] = 1 - E[\lambda_t(t_i)]$. If t_i and t have the same ticket type, we can assume that a new ticket t and historical ticket t_i are independent, i.e., $P[c(t), c(t_i)] = P[c(t)] \cdot P[c(t_i)]$. Then, the expected penalty is

$$E[\lambda_t(t_i)] = \sum_{c(t), c(t_i) \in 0, 1} P[c(t)] \cdot P[c(t_i)] \cdot \lambda_t(t_i).$$

Since $c(t_i)$ is already fixed, substituting $\lambda_t(t_i)$, we obtain

$$E[\lambda_t(t_i)] = \begin{cases} P[c(t) = 0] \cdot (1 - \lambda), & t_i \text{ real ticket} \\ P[c(t) = 1] \cdot \lambda, & t_i \text{ false ticket} \end{cases}$$

Note that all factors in the new similarity function are of the same scale, i.e., [0, 1], thus $0 \le sim'(\cdot, \cdot) \le 1$.

Given an incoming ticket t, the probabilistic fusion method needs to estimate the distribution of $P[c(t)]$. The dividing method also has to predict whether t is a real ticket or a false ticket. There are many binary classification algorithms for estimating $P[c(t)]$. In our implementation, we utilize another KNN classifier. The features are the event attributes and the classification label is the ticket type. The KNN classifier first finds the K nearest tickets in D, denoted as $D_K = \{t_{j_1}, \ldots, t_{j_k}\}$. Then, $P[c(t) = 1]$ is the proportion of real tickets in D_K and $P[c(t) = 0]$ is the proportion of false tickets in D_K.

4.2.4 Representation of Monitoring Tickets

As shown in Sect. 4.2.1, attribute level features are used in the traditional KNN algorithm for recommendation. However, attribute-level feature representation is not interpretable and often contains a lot of noise.

Our observation indicates that each monitoring ticket describes the existing problems (e.g., low capacity, high CPU, utilization) in service, and the associated ticket resolution should be highly relevant to the problems. Table 5 presents some sample monitoring tickets to demonstrate that it is better to use features semantically capturing these problems, instead of attribute-level features, to represent monitoring tickets.

In this section, we describe an application of Latent Dirichlet Allocation (LDA) to feature extraction, which allows us to first extract hidden topics and then to encode monitoring tickets using topic level features. LDA is a generative probabilistic model of a document corpus. Its basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [25]. Following [25], LDA assumes the following generative process for each document w in a corpus D of length M:

1. Choose $\theta \sim Dir(\alpha)$, where $Dir(\alpha)$ is the *Dirichlet distribution* for parameter α
2. For each of the N words w_n:

    (a) Choose a topic $z_n \sim Multinomial(\theta)$.
    (b) Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

According to the graphical model, the total probability $P(D|\alpha, \beta)$ of a corpus D is given by

$$\prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{d_n}} p(z_{d_n|\theta_d}) p(w_{d_n})|z_{d_n}, \beta \right) d\theta_d \qquad (17)$$

Learning the various distribution (the set of topics, their associated word probabilities, the topic of each word, and the topic probabilities of each document) is a

Table 5 Tickets for explaining motivation of incorporating resolution information

Ticket	Summary	Resolution
1	The logical disk has a low amount of free space. Percent available: 2 Threshold: 5	After deleting old uninstall files, the logical disk has now over 10% of free disk space
2	The percentage of used space in the logic disk is 90%. Threshold: 90%	After deleting old uninstall files, the logical disk has now over 15% of free disk space
3	File system is low. The percentage of available space in the file system is 10%. Threshold: 90%	After delprof run, the server now has more than 4gb of free space
4	The logical disk has a low amount of free space. Percent available: 3 Threshold: 5	No trouble was found, situation no longer persists

problem of Bayesian Inference [25]. Topic probability distribution of a document is commonly used as its feature vector.

Following are steps for using LDA for feature extraction in our work:

- Represent each monitoring ticket as a document by concatenating each attribute after removing stop words and introducing tokenization.
- Using historical tickets to train an LDA model.
- Inference feature vectors using the trained LDA model for both incoming events and historical monitoring tickets.

After those steps, monitoring tickets can be encoded as feature vectors and the cosine similarity can then be applied to measure their similarities. Experiments in Sect. 4.3 demonstrate that the algorithm performance based on topic level features is better than that on attribute level features.

### 4.2.5 Incorporating the Resolution Information

In a K nearest neighbor search two types of historical tickets should be ranked higher than others: those with resolutions that are highly relevant to an incoming event and those with resolutions that are more prevalent. Table 5 presents four tickets to demonstrate this point: the resolution from Ticket 1 should have a higher rank than the one from Ticket 4 since the resolution from Ticket 1 is more informative. Similarly, resolutions from Ticket 1 and Ticket 2 should have higher ranks than the one from Ticket 3 because of their higher prevalence.

In Sect. 4.2.1, $sim(e, e(t_i))$ is computed to find the K nearest neighbors of an incoming event e, in which $e(t_i)$ is the event information associated with the i-th ticket. To incorporate the resolution information, $sim(e, t_i)$ (i.e., similarity between an incoming event and the i-th ticket), rather than $sim(e, e(t_i))$, is used in the algorithm. $sim(e, t_i)$ can be easily computed since e and t_i can be vectorized with the same dimensions after using topic-level features. Experiments in Sect. 4.3 demonstrate the effectiveness of this proposed approach.

4.2.6 Metric Learning

According to our observation, topics extracted from the LDA model should have different contributions to the similarity measurement since some topics contain the major descriptive words about events while the others may consist of less meaningful words. For example, Topic A (server wsfpp1 lppza0 lppzi0 nalac application) contains more descriptive words than Topic B (server hung condition responding application apps), and thus should be assigned a larger weight. We adopt metric learning [26] to achieve this goal.

The metric learning [27] problem aims at learning a distance function tuned to a particular task, and has been shown to be useful when used in conjunction with nearest-neighbor methods and other techniques that rely on distances or similarities [28] as illustrated in Fig. 9. To facilitate the learning process, in metric learning, we use a slightly modified form of Mahalanobis Distance function [26]:
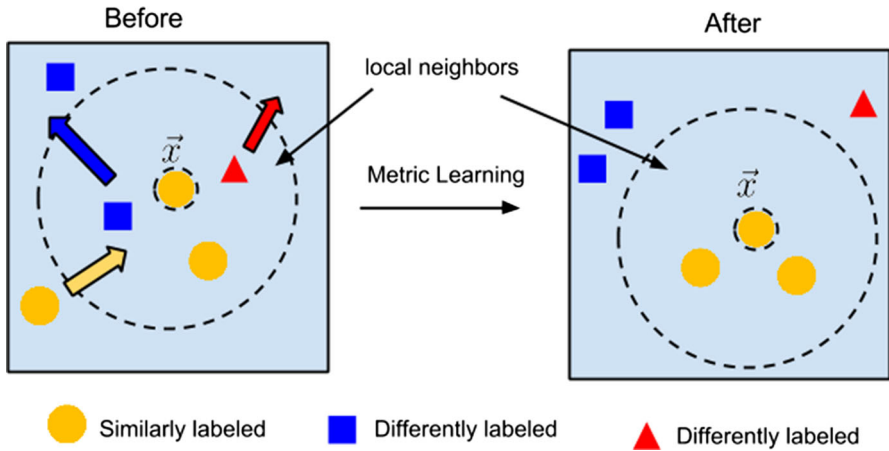
Fig. 9 Before metric learning, vector *x* neighbors include instances from different classes; after metric learning, its neighbors are all belong to the same class

$$d_A(x, y) = x^T A y. \tag{18}$$

In our work, we have n historical tickets $t_1, t_2, \ldots, t_n$ and n corresponding resolutions $r(t_1), r(t_2), \ldots, r(t_n)$. We consider the resolution categories as supervision for metric learning since intuitively similar resolutions solve similar issues. We pre-calculate matrix $R \in R^{n*n}$ in which $R_{i,j} = sim(r(t_i), r(t_j))$. Our goal is to learn a similarity function $S_A(\mathbf{t_i}, \mathbf{t_j})$ by solving the following optimization problem:

$$f(A) = min \sum_{i=1}^{n} \sum_{j=1}^{n} ||R_{i,j} - S_A(\mathbf{t_i}, \mathbf{t_j})||^2 = min||R - SAS^T||^2, \tag{19}$$

in which we use $S_A(\mathbf{t_i}, \mathbf{t_j}) = \mathbf{t_i}^T * A * \mathbf{t_j}$ ($\mathbf{t_i}$ and $\mathbf{t_j}$ are feature vectors for ticket t_i and t_j) instead of $S_A(\mathbf{e}(\mathbf{t_i}), \mathbf{e}(\mathbf{t_j}))$ as we want to keep the benefits of incorporating the resolution information into K nearest search. Since matrix A is constrained to be a positive semi-definite (PSD) matrix, the projected gradient descent algorithm can be directly applied to solve the optimization problem in Eq. (19). In each iteration of gradient descent, the new updated matrix A will be projected into a PSD matrix as the initial value for the next iteration. The singular value thresholding [29] has been applied to project A into a PSD matrix by setting all A's negative eigenvalues to be zero.

The following is the gradient for Eq. (19):

$$\nabla f(A) = \nabla_A((R - SAS^T)^T(R - SAS^T)) = 2S^T SAS^T S - 2S^T AS$$

The resolution categories are usually provided by system administrators. With the available category information, the similarity between two resolutions is 1 if $r(t_i), r(t_j)$ have the same category, and 0 otherwise.

4.3 Evaluation

We show our experimental results from two perspectives. We first compare experimental results from WKNN, Division, and Probabilistic Fusion to show the effectiveness of our proposed methods in eliminating misleading resolutions, and then compare experimental results from LDABaselineKNN, CombinedLDAKNN and MLCombinedLDAKNN to show the efficiency of our proposed methods in improving recommended resolutions' relevance. We still show experimental results between WKNN and LDABaselineKNN since they prove that topic level features do not cause information loss compared to attribute level features. The LDABaselineKNN algorithm is the baseline for CombinedLDAKNN, which itself is the baseline for MLCombinedLDAKNN. We use the Weighted KNN algorithm as the underlying algorithm because it is the most widely used Top-N item-based recommendation algorithm.

Experimental monitoring tickets are collected from three accounts managed by IBM Global Services, denoted later "account1," "account2" and "account3." To evaluate metric learning, 1000 labeled tickets with resolution categories are obtained from "account1".

The following evaluation measures are used in our experiments.

Weighted Accuracy For each ticket set, the first 90% tickets are used as the historic tickets and the remaining 10% tickets are used for testing. Hit rate is a widely used metric for evaluating the accuracy in item-based recommendation algorithms [30–32]. Here we define a recommended resolution as a *hit* if it has Jaccard similarity greater than a *threshold* with the ground truth resolution.

Since real tickets are more important than false ones, we define another accuracy measure, the weighted accuracy that assigns weights to real and false tickets. In this evaluation, the importance weight of the real tickets λ is 0.9 since the real tickets are much more important than the false tickets in reality. We also test other large λ values, such as 0.8 and 0.99. The accuracy comparison results have no significant change. As shown by Fig. 10a, our proposed two algorithms have smaller penalties than the traditional KNN-based recommendation algorithms. The probabilistic fusion method outperforms the division method, which relies heavily on the ticket type predictor. Overall, our probabilistic fusion method only has about one-third of the penalties of the traditional.
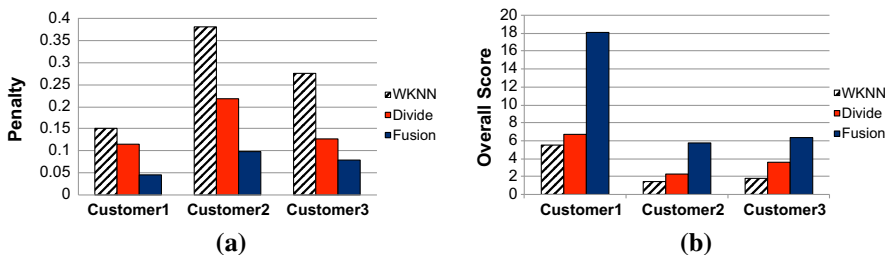

Fig. 10 **a** Average penalty and **b** overall score for $K = 10$ and $k = 3$

Average Similarity In general, several resolutions can be recommended for a single testing instance. To consider the relativeness of all recommended resolutions, the *average similarity* (avgSim) is used as one evaluation metric that is given by the following equation:

$$\text{avgSim} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{n_i} sim(r_{io}, r_j)/n_i,$$

in which N is the number of testing instances, and n_i is the number of recommended resolutions for testing instance i and r_{io} is its original resolution, and r_j is its jth recommended resolution. Jaccard Similarity is used to calculate $sim(r_{io}, r_j)$.

Mean Average Precision We also assess Mean Average Precision (MAP) [33] which is widely used for recommendation evaluation.

Evaluation on eliminating misleading resolutions An overall quantity metric is used for evaluating the recommendation algorithms, covering both the accuracy and the average penalty. It is defined as the overall score = weighted accuracy/average penalty. If the weighted accuracy is higher or the average penalty is lower, then the overall score becomes higher and the overall performance is better. Figure 10b shows the overall scores of all algorithms for two parameter settings. It is seen that our proposed algorithms are always better than the weighted KNN algorithm in each data set.

To compare the results of each algorithm, we vary the number of recommendation resolutions, k. When we increase the value of k, the size of the recommendation results becomes larger. Then the probability of one recommended resolution being *hit* by the true resolution also increases. Therefore, the weighted accuracy becomes higher. Except for the division method, all algorithms have similar weighted accuracies for each k. However, as k increases and there are more recommended resolutions, there are more potential penalties in the recommended resolutions. Hence, the average penalty also becomes higher. Clearly, the probabilistic fusion method outperforms other algorithms for every k.

For other values of K ranging from 8 to 20, the comparison results are very similar to $K = 10$. Usually, we set $K = 10$ and $k = 5$ in practice. The choice of k is a tradeoff between accuracy and a reasonable number of recommended resolutions. Large k decreases user experience since system administrators have to choose a proper one out of candidate resolutions, meanwhile small k greatly decreases accuracy.

We select an event ticket in "account1" to illustrate why our proposed algorithms are better than the traditional KNN-based algorithms. Table 6 shows a list of recommended resolutions given by each algorithm. The testing ticket is a real event ticket triggered by a low capacity alert for the file system. Its true resolution of this ticket is as follows: "cleaned up the FS using RMAN retention policies..." RMAN is a data backup and recovery tool in the Oracle database. The general idea of this resolution is to use this tool to clean up the old data.

As shown in Table 6, the first resolution recommended by WeightedKNN is a false ticket's resolution: "No actions were taken by GLDO for this Clearing Event..." It might be caused by a temporal file generated by some application,

**Table 6** A Case Study for $K = 10$ and $k = 3$

Algorithm	Recommended resolution	Is hit	Is real ticket's resolution	Penalty
Weighted KNN	No actions were taken by GLDO for this Clearing Event...	No	False	0.9
	I cleaned up the FS using RMAN retention policies...	Yes	True	0
	Duplicated 28106883...	No	True	0
Divide	Duplicated 28106883...	No	True	0
	Another device failure has been reported for this node...	No	True	0
	I cleaned up the FS using RMAN retention policies...	Yes	True	0
Fusion	Duplicated 28106883...	No	True	0
	Another device failure has been reported for this node...	No	True	0
	I cleaned up the FS using RMAN retention policies...	Yes	True	0

which would clean up the temporal file automatically after its job was done. When the system administrator opened that ticket, the problem was gone, and that ticket is seen as false. However, the testing ticket is real and would not disappear unless the problem was actually fixed. This resolution from the false ticket would have misled the system administrator to overlook this problem. Consequently, a penalty of $\lambda = 0.9$ is given to WeightedKNN.

WeightedKNN, Divide and Fusion all successfully find the true resolution of this testing ticket, but WeightedKNN has one false resolution, so its penalty is 0.9. Our proposed methods, Divide and Fusion, have no penalty for this ticket. Therefore, the two methods are better than WeightedKNN.

Evaluation on Improving Accuracy Figure 11a shows the experimental results of choosing the proper number of topics for training the LDA model using data set "account1". *numTopics* = 300 is a proper setup for the number of topics. Thus, we choose *numTopics* = 300 for all the following experiments. The *average similarity* is used for comparing the performance among WKNN, LDABaselineKNN and CombinedLDAKNN. When resolution categories are available, *MAP@n* is used for comparing the performance between CombinedLDAKNN and MLCombinedLDAKNN since it explicitly considers the relativeness of the recommended results. To compare the results of each algorithm, we vary the number of recommended resolutions, k. Topic level features are better than attribute level features for account1 and account2 and slightly worse for account3 by comparing algorithm WKNN and LDABaselineKNN. CombinedLDAKNN always outperforms LDABaselineKNN, which proves the effectiveness of incorporating the resolution information into the K nearest neighbor search.

Metric Learning Performance Our analysis of the metric learning performance showed that the similarity scores between monitoring tickets with resolutions from
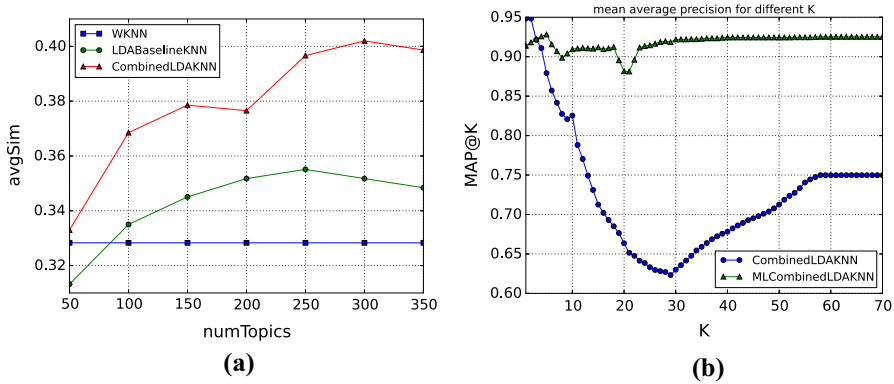
Fig. 11 **a** Accuracy varies for different *numTopics* for dataset "account1" and **b** mean average precision varying parameter K of underlying KNN algorithm

the same category are enhanced while similarity scores between monitoring tickets with resolutions from different categories are reduced.

As shown in Fig. 11b, the overall MAP scores of MLCombinedLDAKNN are also higher and more stable than CombinedLDAKNN when K increases. It indicates that MLCombinedLDAKNN can retrieve more related resolutions first and thus is more robust to noisy resolutions compared to CombinedLDAKNN, which proves the effectiveness of metric learning.

4.4 Lessons Learned

In this section, we describe an example of building autonomic systems for IT service management where the machine learning approach is used to determine event resolution. The approach improves the peroformance of the KNN algorithm by utilizing division and probabilistic fusion methods, as well as LDA for adding topic level features combined with metric learning for added precision when data are available. The complexity of service environments and limitations of the historical data captured manually during delivery of the services, creates a necessity for addressing often overlooked details such as the existence of False Tickets and the absence of full detailed descriptions. However, as our experiments on the real data demonstrated, the approaches described in this section are effective and efficient in an automated resolution recommendation for monitoring tickets.

5 Related Work

Our work on building autonomic systems for workload management in IT service systems is inspired by a class of workload management methods based on *control theory* [3]. Various control-theoretic approaches have been used for performance management of IT infrastructure from studying the behavior of dynamic systems and feedback-driven control. Diao et al. [34] proposes a feedback control based

approach to automated enforcement of service level agreements. Padala et al. [35] develops an adaptive resource controller to meet application-level quality of service goals in a data center environment. Lightstone et al. [36] studies the use of control theory for database workload management. However, none of the above feedback approaches have been applied to workload management in service delivery organizations.

Our work on building autonomic systems for workforce management in IT service systems is close to the so-called Skills Based Routing ("SBR"). The SBR problem is known to be analytically complex with limited theoretical results. Aksin et al. [37] provides detailed surveys of the analytical approaches that have been undertaken. Common approaches are to simplify the topology of the network or simplify the routing schemes. However, these are not desirable solutions in a service delivery environment where both the network and the routing schemes are complex and the service providers are seeking practical solutions rather than conceptual guidance. An alternative solution methodology that has applied to the SBR problem is a simulation-based approach. Atlason et al. [38] considers a multi-period problem of determining optimal staffing levels and solving a sample average approximation of the problem using a simulation based analytic center cutting plane method. Cezik et al. [39] extends this approach by applying it to large problem instances and developing heuristic methods to handle the numerical challenges that arise. However, while vast literature exists for solving the SBR problem in the context of manufacturing systems and call centers, very little research has been conducted in service delivery systems due to the complexity of the customer workload. On the other hand, there are various non-SBR problems solved in a service delivery system related to the management of incidents, problems, and changes. Zia et al. [40] proposes a change scheduling optimization model that can be solved using mixed integer programming. Diao et al. [41] studies a rule-based approach for problem ticket classification.

A substantial amount of research has been devoted to the recommendation systems. The existing recommendation algorithms can be categorized into two types. The first type is a learning-based recommendation, in which the algorithm aims to maximize the rate of user response, such as user click or conversation. The recommendation problem is then naturally formulated as a prediction problem. It utilizes a prediction algorithm to compute the probability of the user response on each item. Then, it recommends the one having the greatest probability. Most prediction algorithms can be utilized in the recommendation, such as naive Bayes classification, linear regression, logistic regression and matrix factorization [42]. The second type of recommendation algorithm focuses on the relevance of items or users, rather than the user response. Many algorithms proposed for promoting products to online users [43] belong to this type. They can be categorized as item-based algorithms (e.g., [21]) and user-based algorithms (e.g., [44]). Metric learning partially overcomes the difficulties of feature extraction and similarity measurement in those domains. Metric learning requires learning a distance metric for the input space of data from a given collection of a pair of similar/dissimilar points that preserves the distance relation among the training data. In addition, many machine learning algorithms, such as KNN, heavily rely on the underlying distance metric

for the input instances. Previous work such as [45] has shown that appropriately designed distance metrics can significantly benefit KNN-based algorithms compared to the standard Euclidean distance.

6 Conclusions and Future Work

Enterprises and service providers are increasingly challenged with improving the quality of service delivery while containing the cost. However, it is often difficult to effectively manage the complex relationships among dynamic customer workloads, strict service level requirements, and efficient service management processes. In this paper, we presented our progress on building autonomic systems for IT service management through a collection of automated data driven methodologies. This includes the design of feedback controllers for workload management, the use of simulation-optimization methodology for workforce management, and the development of machine learning models for service event management. We demonstrated the applicability of the presented approaches using examples and data from a large IT services delivery environment.

IT service management is a rich research field with complex problems delivering significant values. Despite the progress presented in this paper, building automated data driven systems for IT service management still encompasses abundant opportunities until it reaches its full potential. Continuous learning and optimization will be at the core of building automated solutions that will shift IT service management from mostly reactive to proactive modes of operation. The drive for greater speed, quality, and consistency of IT service management, coupled with the need to manage very complex environments at scale, has also called for an integrated system that can incorporate deep analytics technologies over vast amount of data with aggregated data sources across the whole service management lifecycle. Such a continuous learning and integrated system will enhance the relationship between service customers and service providers, shifting towards an agile co-creation of automated data driven systems that achieve higher service quality with reduced management cost.

References

1. Office of Government Commerce: IT Infrastructure Library. ITIL Service Support, version 3 (2007)
2. Diao, Y., Heching, A.: Workload management in dynamic IT service delivery organizations. In: Proceedings of International Workshop on Distributed Systems: Operations and Management, Venice, Italy (2009)
3. Hellerstein, J.L., Diao, Y., Parekh, S., Tilbury, D.M.: Feedback Control of Computing Systems. Wiley, Hoboken (2004)

4. Astrom, K.J., Wittenmark, B.: Adaptive Control, 2nd edn. Addison-Wesley Publishing Company, Reading (1994)
5. Franklin, G.F., Powell, J.D., Emani-Naeini, A.: Feedback Control of Dynamic Systems, 3rd edn. Addison-Wesley, Reading (1994)
6. Fleming, W.H., Rishel, R.W.: Deterministic and Stochastic Optimal Control. Springer, Berlin (1996)
7. Lavenberg, S.S. (ed.): Computer Performance Modeling Handbook. Academic Press, INC, Orlando (1983)
8. Diao, Y., Heching, A.: Staffing optimization in complex service delivery systems. In: Proceedings of 7th International Conference on Network and Service Management (2011)
9. Hellerstein, J.L., Diao, Y., Parekh, S.: A first-principles approach to constructing transfer functions for admission control in computing systems. In: Proceedings of IEEE International Conference on Decision and Control (2002)
10. Diao, Y., Hellerstein, J.L., Parekh, S.: Stochastic modeling of lotus notes with a queueing model. In: Proceedings of International Computer Measurement Group Conference (2001)
11. Diao, Y., Hellerstein, J.L., Parekh, S., Shaikh, H., Surendra, M., Tantawi, A.: Modeling differentiated services of multi-tier web applications. In: Proceedings of IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (2006)
12. Diao, Y., Heching, A., Northcutt, D., Stark, G.: Modeling a complex global service delivery system. In: Proceedings of 2011 Winter Simulation Conference, Phoenix, Arizona, pp. 690–702 (2011)
13. Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H., Zeltyn, S., Zhao, L.: Statistical analysis of a telephone call center: a queueing-science perspective. J. Am. Stat. Assoc. 100, 36–50 (2005)
14. Ulrich, R., Miller, J.: Information processing models generating lognormally distributed reaction times. J. Math. Psychol. 37, 513–525 (1993)
15. Glover, F., Laguna, M., Mart, R.: Scatter search. In: Ghosh, A., Tsutsui, e S (eds.) Advances in Evolutionary Computation: Theory and Applications, pp. 519–537. Springer, New York (2003)
16. Glover, F.: Tabu search. Decis. Sci. 8, 156–166 (1977)
17. Karpov, Y.G.: Anylogic: a new generation professional simulation tool. In: VI International Congress on Mathematical Modeling. NizniNovgorog, Russia (2004)
18. XJ Technologies: http://www.xjtek.com// (2011)
19. Zhou, W., Tang, L., Zeng, C., Li, T., Shwartz, L., Grabarnik, G.Y.: Resolution recommendation for event tickets in service management. IEEE Trans. Netw. Serv. Manag. 13(4), 954–967 (2016)
20. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. 46(3), 175–185 (1992)
21. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T.: Application of dimensionality reduction in recommender system—a case study. In: ACM WebKDD Workshop (2000)
22. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison Wesley, Reading (2005)
23. Manning, C.D., Raghavan, P., Schütze, H., et al.: Introduction to Information Retrieval, vol. 1. Cambridge University Press, Cambridge (2008)
24. Tang, L., Li, T., Pinel, F., Shwartz, L., Grabarnik, G.: Optimizing system monitoring configurations for non-actionable alerts. In: IEEE/IFIP Network Operations and Management Symposium (NOMS) (2012)
25. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022 (2003)
26. Kulis, B.: Metric learning: a survey. Found. Trends Mach. Learn. 5(4), 287–364 (2012)
27. Aurlien, B., Amaury, H., Marc, S.: Metric Learning Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, San Rafael (2015)
28. Frome, A., Singer, Y., Sha, F., Malik, J.: Learning globally-consistent local distance functions for shape-based image retrieval and classification. In: Computer Vision, ICCV 2007. IEEE, pp. 1–8 (2007)
29. Cai, J.-F., Candès, E.J., Shen, Z.: A singular value thresholding algorithm for matrix completion. SIAM J. Optim. 20(4), 1956–1982 (2010)
30. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst. 22(1), 143–177 (2004)
31. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: CIKM, pp. 247–254 (2001)
32. Ning, X., Karypis, G.: SLIM: Sparse linear methods for top-n recommender systems. In: ICDM, pp. 497–506 (2011)

33. Zhu, M.: Recall, precision and average precision. Department of Statistics and Actuarial Science, University of Waterloo, Waterloo (2004)
34. Diao, Y., Hellerstein, J.L., Parekh, S.: A business-oriented approach to the design of feedback loops for performance management. In: Proceedings of International Workshop on Distributed Systems: Operations and Management, Nancy, France (2001)
35. Padala, P., Shin, K.G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A., Salem, K.: Adaptive control of virtualized resources in utility computing environments. In: Proceedings of the 2007 EuroSys Conference, Lisbon, Portugal (2007)
36. Lightstone, S.S., Surendra, M., Diao, Y., Parekh, S., Hellerstein, J.L., Rose, K., Storm, A.J., Garcia-Arellano, C.: Control theory: A foundational technique for self managing databases. In: Proceedings of IEEE International Conference on Data Engineering (Workshop) (2007)
37. Aksin, Z., Armony, M., Mehrotra, V.: The modern call center: a multi-disciplinary perspective on operations management research. Prod. Oper. Manag. **16**, 665–688 (2007)
38. Atlason, J., Epelman, M.A., Henderson, S.G.: Optimizing call center staffing using simulation and analytic center cutting-plane methods. Manag. Sci. **54**, 295–309 (2008)
39. Cezik, M.T., LaEcuyer, P.: Staffing multiskill call centers via linear programming and simulation. Manag. Sci. **54**, 310–323 (2008)
40. Zia, L., Diao, Y., Rosu, D., Ward, C., Bhattacharya, K.: Optimizing change request scheduling in IT service management. In: Proceedings of IEEE International Conference on Services Computing (2008)
41. Diao, Y., Jamjoom, H., Loewenstern, D.: "Rule-based problem classification in IT service management. In: Proceedings of IEEE International Conference on Cloud Computing (2009)
42. Manning, C.D., Schuetze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)
43. Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: ICDM, pp. 43–52 (2007)
44. Terveen, L., Hill, W.: Beyond recommender systems: Helping people help each other. In: HCI in the New Millennium, pp. 487–509 (2001)
45. He, X., King, O., Ma, W.-Y., Li, M., Zhang, H.-J.: Learning a semantic space from user's relevance feedback for image retrieval. IEEE Trans. Circuits Syst. Video Technol. **13**(1), 39–48 (2003)

Yixin Diao is a Research Staff Member at IBM T. J. Watson Research Center. He received his Ph.D. degree in Electrical Engineering from Ohio State University. He has published more than eighty papers in systems and services management and is the co-author of the book "Feedback Control of Computing Systems." He has received Best Paper Awards from IEEE/IFIP Network Operations and Management Symposium, IFAC Engineering Applications of Artificial Intelligence, and IEEE International Conference on Services Computing. He is an Associate Editor of IEEE Transactions on Network and Service Management, and Journal of Network and Systems Management. He is a Fellow of IEEE.

Larisa Shwartz is a Senior Technical Staff Member at IBM T. J. Watson Research Center. She received her Ph.D. degree in Mathematics from UNISA University. Her research experience includes mathematics and computer science with current focus on analytics for IT service management, cloud systems, and automation. She has over 50 publications and more than 80 patents and patent applications. She is on the editorial board of International Journal of Analytics and serves as TPC member of various conferences such as ICSOC, SIMUL, and SOCA. She is a recipient of IBM Corporate Award in 2017 and a number of IBM Research awards including Outstanding Technical Innovation Award in 2015.