Efficient Ticket Routing by Resolution Sequence Mining

Qihong Shao, Yi Chen
Arizona State University
{qihong.shao, yi}@asu.edu

Shu Tao, Xifeng Yan, Nikos Anerousis
IBM T. J. Watson Research Center
{shutao, xifengyan, nikos}@us.ibm.com

ABSTRACT

IT problem management calls for quick identification of resolvers to reported problems. The efficiency of this process highly depends on ticket routing—transferring problem ticket among various expert groups in search of the right resolver to the ticket. To achieve efficient ticket routing, wise decision needs to be made at each step of ticket transfer to determine which expert group is likely to be, or the one that can lead to the resolver.

In this paper, we address the possibility of improving the efficiency of ticket routing by mining ticket resolution sequences alone, without accessing ticket content. To demonstrate this possibility, a Markov model is developed to statistically capture the right decisions that have been made toward problem resolution, where the order of the Markov model is carefully chosen according to the conditional entropy obtained from ticket data. We also design a search algorithm, called Variable-order Multiple active State search (VMS), that generates ticket transfer recommendations based on our model. The proposed framework is evaluated on a large set of real-world problem tickets. The results demonstrate that VMS significantly improves human decisions: the problem resolver can often be identified with fewer ticket transfers.

1. INTRODUCTION

Problem management is a key task in managing today's enterprise computing environment, a multi-billion-dollar business. Its goal is to quickly resolve the reported problems, e.g., hardware failures, software bugs, application errors, etc., hence minimize the disruptions caused to business operations. Problem management is typically conducted by the helpdesk and IT support staff members of an enterprise.

The process of problem management is reflected by the lifecycles of problem tickets. A ticket is opened as soon as a problem is reported. Then, it is routed among various expert groups until the root cause of the problem is identified by a *resolver group*. Finally, the resolver group solves the problem and closes the ticket. Table 1 shows a sample ticket lifecycle, in which the ticket was routed among multiple groups before it was solved. The efficiency of identifying the problem resolver depends on ticket routing, which decides, if the group currently holding the ticket cannot solve the problem, which group the ticket should be transferred to next[1].

ID	Time	Entry
28120	2007-05-14	New Ticket: DB2 login failure
28120	2007-05-14	Transferred to Group SMRDX
28120	2007-05-14	Contacted Mary for recycling WAS
28120	2007-05-14	Transferred to Group SSDSISAP
28120	2007-05-14	Status updated ...
28120	2007-05-15	Transferred to Group ASWWCUST
28120	2007-05-15	Web service checking
28120	2007-05-18	Could not solve the problem.
...	...	...
28120	2007-05-18	Transferred to Group SSSAPHWOA
28120	2007-05-22	Resolved

Table 1: A sample ticket lifecycle

Today, ticket routing is usually driven by expert decisions. It is not uncommon that due to human error or inexperience a ticket is mistakenly transferred to a group that cannot solve the problem, which might lead to a long and inefficient routing sequence. In such cases, not only resources are wasted, but also it would take longer time to close tickets, causing customer dissatisfaction.

In this paper, we propose a mechanism that can improve the overall efficiency of ticket routing, measured by the Mean number of Steps To Resolve (MSTR) the tickets. The first question is what information in ticket data can be used to improve MSTR. Typically, a problem ticket contains two types of information: (1) ticket content that includes problem description and diagnostic data, (2) resolution sequence that shows how it was routed before it reached the resolver group. In the sample ticket shown in Table 1, the entries compose the ticket content, while the extracted group names ⟨SMRDX, SSDSISAP, ASWWCUST, SSSAPHWOA⟩ form its resolution sequence. Our study in this paper focuses on the resolution sequences only. As shown later, mining the resolution sequences alone can significantly improve the efficiency of ticket routing and identify resolver groups more quickly – a surprisingly encouraging discovery.

Our strategy is to mine resolution sequences of solved tick-

[1]Due to the cost of problem diagnosis, a "broadcast" scheme that sends tickets to all expert groups simultaneously is too expensive in practice.

ets, which could guide the routing decisions for new tickets. With in-depth analysis, we find that in many cases, long resolution sequences were results of a few local mis-routing decisions, while the majority of the local ticket transfer decisions were logically correct. Intuitively, for a specific type of problems, these local ticket transfer decisions reflect the functional relationships between expert groups. For instance, if group A often transfers `AIX` tickets to group B, this implies if A cannot resolve an `AIX` problem, B is very likely to be able to solve it. Based on this intuition, we build a model to capture these relationships by mining transfer decisions recorded in the solved tickets. The model is then applied to guide future ticket transfer toward correct decisions.

Our method is based on Markov models (or Markov chains). Let each Markov state represents a group, the transition probabilities between these states capture the local decisions, i.e. the likelihood of a group to be a transfer target, given the previous groups that have processed the ticket. There are several challenges in applying Markov model in this problem, e.g., what kind of group transitions should be captured by the model? We can take the transitions from the previous groups to the resolver group, or use intermediate group transitions? what should be the optimal order of the derived Markov model? How should the model be used to guide ticket transfer? In our study, we investigate these issues and design a search algorithm that generates good ticket routing recommendations based on learned models.

In summary, our contributions in this paper include:

- We successfully exploit the potential of mining resolution sequence to improve ticket routing. To our best knowledge, this is the first study on this problem from a data mining perspective.
- We develop a Markov model with variable-order to statistically capture the ticket transfer decisions that have been made toward problem resolution, where the order of the Markov model is set according to the conditional entropy estimated from ticket data to boost accuracy. We then design an algorithm, called Variable-order Multiple active State search (VMS), which leverages the developed model to best predict the problem resolver.
- We conduct extensive experiments to verify our approach and demonstrate that VMS can effectively shorten ticket resolution sequences. Thus, it can be adopted as a recommendation tool for ticket routing.

The remainder of this paper is organized as follows. We first formulate the problem in Section 2. Then we present the proposed Markov model that captures the ticket transfer decisions in Section 3, and algorithms that search for resolver groups based on the derived model in Section 4. In Section 5, we evaluate the effectiveness and robustness of the proposed approach. The related works are reviewed in Section 6. Finally, Section 7 concludes the paper and discusses future directions of this study.

2. PROBLEM FORMULATION

A problem ticket can be represented by a tuple with two components, (τ, s), where τ is the ticket content and s is the routing sequence. A ticket is routed to find the group that can potentially solve the problem. Let $\mathcal{G} = \{g_1, g_2, \ldots, g_n\}$ be the set of all expert groups. The routing sequence of a ticket can be written as $G_{(k)} = \langle g_{(1)}, g_{(2)}, \ldots, g_{(k)} \rangle$ ($g_{(i)} \in \mathcal{G}$), in which a ticket is first issued to $g_{(1)}$, then transferred in the order of $g_{(2)}, g_{(3)}, \ldots, g_{(k)}$. A *step* in $G_{(k)}$ is a ticket transfer from one group to another. A ticket $(\tau, G_{(k)})$ is open if none of the groups in $G_{(k)}$ can resolve it. Correspondingly, a ticket is closed if the last group in $G_{(k)}$, i.e., $g_{(k)}$, solved the problem, and in this case, the routing sequence is called a *resolution sequence*.

Given m ticket resolution sequence $\{G_i\}_{i=1,\ldots,m}$, we measure the efficiency of a ticket routing system using the Mean number of Steps To Resolve (MSTR):

$$T = \frac{\sum_{i=1}^{m} |G_i|}{m}. \quad (1)$$

Since the more steps involved in ticket resolution, the longer delay it will cause for the ticket to be closed, our goal is to minimize Eq. (1).

We approach this problem by mining the ticket resolution sequences $G_{(k)}$ and address the following question:

Is it possible to reduce MSTR by mining ticket resolution sequence $G_{(k)}$ alone without accessing ticket content τ?

As we show later in the paper, although seemingly counter-intuitive, the answer is positive. Why? Because if two tickets share the similar resolution process in the past, likely they are related to similar problems. As a result, they might share the similar resolution route toward the end!

Formally, given a ticket resolution history database, $D_S = \{G_1, G_2, \ldots, G_m\}$, we develop a framework to mine the resolution sequences in D_S and make routing recommendation on open tickets in a testing ticket dataset T_S in order to minimize their MSTRs. For tickets in T_S, our system is only provided with initial routing information, i.e., the first group the ticket was assigned to.

3. RESOLUTION SEQUENCE MINING

In this section, we introduce the Markov model that we use to capture the ticket transfer decisions embedded in ticket resolution sequences.

3.1 Modeling Ticket Transfer Decisions

Markov models are widely used to capture the temporal dependencies between the states of a system. A process is considered Markovian if at any time point, the probability of the process in the current state is solely dependent on its previous states. Given such property, Markov model fits naturally with our problem here. In our problem setting, each group can be modeled as a state. Assume there are N expert groups, $\mathcal{G} = \{g_i\}, i \in \{1, 2, \ldots, N\}$, to which a ticket can be transferred. A kth-order Markov model gives the probability of the ticket being transferred to group g_i at the next step, $P(g = g_i | G_{(k)}), g_i \in \mathcal{G}$.

In ticket routing, decisions are often based on past group transfers irrespective of the order of transfer. In other words, the decision maker typically looks at all history updates generated by the past groups, rather than in which order these updates were generated. This observation indicates that, if we use $S_{(k)}$ to denote the set of groups in $G_{(k)}$ (i.e., $S_{(k)} = \{g_{(1)}, g_{(2)}, ..., g_{(k)}\}$), we only need to model $P(g = g_i | S_{(k)}), g_i \in \mathcal{G}$. Therefore, unless otherwise stated, we will refer to this generalized model as our Markov model (in some literatures, this is called quasi-Markov model [14]).

Using historical ticket resolution sequences, we can find the number of instances with a set of group transfers, $S_{(k)}$, denoted as $N(S_{(k)})$, as well as the number of instances that a ticket is transferred to group g_i after processed by $S_{(k)}$, denoted as $N(g_i, S_{(k)})$. We can estimate $P(g_i|S_{(k)})$ by

$$P(g_i|S_{(k)}) = \begin{cases} N(g_i, S_{(k)})/N(S_{(k)}) & \text{if } N(S_{(k)}) > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

$(S_{(k)}, g_i)$ is called a *recognizable group transfer pattern*, which is related to sequential patterns [2] except that it does not have minimum frequency requirement. Note that if the same group transfer pattern occurred multiple times in a single ticket, we only count it once in $N(S_{(k)})$ and $N(g_i, S_{(k)})$. This is to avoid the bias caused by repeated group transfers in a single ticket.

3.2 Do Intermediate Transfer Steps Matter?

An open problem in the above modeling is whether we should build the model using (1) only the ticket transfer instances that led to the resolver group, or (2) all ticket transfer instances in the resolution sequence (i.e., including the intermediate transfer steps).

For case 1, we define $I(\tau, g_i) = 1$ if group g_i can solve problem τ, and 0 otherwise. Let $P(I(\tau, g_i)|\langle g_{(1)}, g_{(2)}, \ldots g_{(k)}\rangle)$ be the probability that problem τ cannot be solved by groups $g_{(1)}, g_{(2)}, \ldots g_{(k)}$, but can be solved by g_i. If we only use the ticket transfer sequences, Eq. (2) is evaluated as

$$P(g_i|S_{(k)}) = E_\tau(I(\tau, g_i)|S_{(k)}). \quad (3)$$

However, as we shall see, the above method does not always yield the best result due to the following two reasons. First, in some cases, it is necessary to transfer tickets to non-resolver or intermediate groups. These groups are typically the "distributor" of the tickets and have a better knowledge of how to match tickets with the expertise of a subset of groups (e.g., groups may be organized in a hierarchy and parent nodes could act as distributors). Second, the probabilities $P(I(\tau, g)|S_{(k)})$ obtained from ticket resolution sequence is robust only if they are built on a sufficiently large number of tickets.

For case 2, we define $J(\tau, g_i) = 1$ if the problem τ should go through group g_i (i.e., the problem can be either solved or correctly routed by g_i), and 0 otherwise. Then, Eq. (2) is evaluated as

$$P(g_i|S_{(k)}) = E_\tau(J(\tau, g_i)|S_{(k)}). \quad (4)$$

In this case, we build our model based on all intermediate transfer steps in ticket resolution sequence. This helps overcome the disadvantage of using Eq. (3) in our modeling. First, if group g_i is a distributor group, this model will be able to capture it, as $P(g_i|S_{(k)})$ would tend to be greater than the transition probabilities to other groups. Second, in cases where no enough ticket data is available to build robust probabilities from Eq. (3), using Eq. (4) instead can result in more training instances, hence can potentially avoid overfitting.

With Eq. (4), we assume that although unwise decisions do exist in the training data, most of the ticket transfer decisions were helpful in identifying the right resolver groups (see experiments in Section 5). If our model accurately captures these decisions, when a new ticket emerges, our model will statistically point us to the right direction of ticket routing. In the following discussion, unless otherwise stated, we will use Eq. (4) to derive our model.

ticket sets	$k=1$	$k=2$	$k=3$	$k=4$	k_{opt}
1	0.172	0.162	0.150	0.149	1
2	0.105	0.104	0.103	0.102	1
3	0.115	0.109	0.109	0.108	1
4	0.062	0.046	0.045	0.043	2
5	0.501	0.454	0.435	0.425	3
6	0.309	0.283	0.280	0.278	2

Table 2: Conditional entropies w.r.t order k

3.3 Optimizing Markov Order

The order of a Markov model determines how many past states are considered to *predict* the future state of the process. To determine the "optimal" order of a Markov model, we consider the conditional entropy of the training data.

In information theory, conditional entropy quantifies the remaining uncertainty of a random variable given the value of a second random variable [5, 8]. In our problem setting, we evaluate the entropy of the next group g conditioned on a given set of past groups $S_{(k)}$, which is denoted as $H(g|S_{(k)})$:

$$H(g|S_{(k)}) = -\sum_{S_{(k)} \in \mathcal{G}^k} P(S_{(k)}) \sum_{g \in \mathcal{G}} P(g|S_{(k)}) \log P(g|S_{(k)}) \quad (5)$$

Here, we define $\log 0 = 0$, as $P(g|S_{(k)})$ could be 0. $\mathcal{G}^k$ stands for all k-group combinations selected from the set of all the groups $\mathcal{G}$.

The conditional entropy $H(g|S_{(k)})$ is 0 if the next group can be fully predicted by the past k groups $S_{(k)}$, and equals $H(g)$ if the next group g is independent of the past group transfers. Obviously, $H(g|S_{(k)})$ is a function of the Markov order k. By evaluating the value of $H(g|S_{(k)})$, we can determine the "optimal" value of k for our Markov model.

Table 2 shows the conditional entropy as the Markov order k varies from 1 to 4, for the Markov models learned from 6 different ticket datasets. As one can see, the conditional entropy can be improved by increasing the Markov order. This suggests that in practice, it is better to make ticket transfer decision based on longer history.

Although increasing Markov order generally leads to better predictability, it also increases the complexity of our model, especially when we apply it to online ticket transfer recommendations (to be discussed in the next Section). Therefore, we need to find a right tradeoff for k. We observe in Table 2 that when k is increased beyond a certain threshold, the improvement of predictability becomes smaller. In our study, we empirically set a threshold $\theta = 0.015$ to determine the optimal value of k, denoted as k_{opt}. Specifically, we consider $k_{opt} = k$ if k is the smallest value that satisfies

$$H(g|S_{(k)}) - H(g|S_{(k+1)}) < \theta. \quad (6)$$

Note that in practice, the optimal order may not be usable due to the limited size of training dataset. To demonstrate this, we plot in Fig. 1 the number of recognizable group transfer patterns with respect to varied Markov order for tickets collected from `AIX` problem during 4 different periods. It is clear that with the increase of Markov order, fewer transfer patterns become available, which may decrease the applicability of our model somehow.

In a k_{opt}-th order Markov model, to measure the proba-

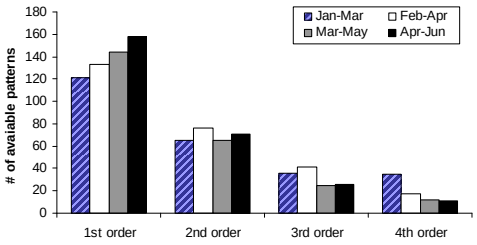


Figure 1: Number of recognizable group transfer patterns vs. Markov order for AIX problem

bility of selecting group g we consider the past k_{opt} groups. If the past k_{opt} group pattern occurs too infrequently in the training dataset (thus the probability becomes statistically unreliable), we will decrease k to $k_{opt} - 1$, $k_{opt} - 2$, and so on until the occurrences of the recognizable k-group patterns become sufficient.

4. TICKET ROUTING ALGORITHMS

As shown in Section 3, our Markov model captures the likelihood that a ticket would be transferred to a group, given the past group transfer information. The next issue is how to use it to make effective ticket routing recommendations, so that a new ticket can be transferred to its resolver group as quickly as possible. Note that the right resolver group for a ticket is unknown at the beginning of ticket routing. What we know is the initial group that a problem ticket was assigned to.

Based on the initial assignment information, we study three algorithms that could guide ticket routing by searching for potential resolver groups using the constructed Markov model. These three algorithms fall into two categories: the first two use 1st-order Markov models, while the third one exploits higher, variable-order models.

4.1 First-order Search Algorithms

The first algorithm, called *First-order Memoryless search* (FM), searches for resolver groups based on a first-order Markov model, i.e., $k = 1$ in Eq. (2). In this algorithm, ticket transfer decision is solely based on the current group. That is, given the current group $g_{(l)}$, the algorithm selects the next group g^*, where

$$g^* = \text{argmax}_g \; P(g|g_{(l)}), \forall g \in \mathcal{G}. \tag{7}$$

Consider the first-order Markov model as a graph, where each node represents a group, and an edge between nodes represents possible ticket transfer. For a new problem ticket, starting from the node representing the initial group, the algorithm traverses the graph and searches for the resolver group in a way similar to depth-first search. At each step, it chooses the next node with the highest transition probability. The search progresses until it finds the resolver, or reaches a node without any unvisited neighbor nodes. In the latter case, FM returns to the most recently visited node whose neighbor nodes have not been fully explored. Here, we assume a ticket should not visit the same group twice.

Example 1: Fig. 2(a) shows a sample 1st-order Markov model, where the value on each edge is the transfer probability between groups, estimated by Eq. (2).

Suppose an incoming ticket is initially assigned to group A and the expected solver group is F. We perform an FM search in Fig. 2(a). Starting from group A, there are four possible next groups $\{C, D, E, G\}$ with transfer probabilities $0.5, 0.45, 0.03$, and 0.02, respectively. Since group A cannot solve the problem, the algorithm selects the group with the highest transition probability 0.5 as the next group, i.e., group C. Following similar steps, the ticket is then transferred to groups B, H, I, J, until it reaches the resolver group F. The search path of this algorithm is marked with thick lines in the figure. The groups that are involved in ticket transfer are shadowed, with a number in parentheses denoting the transfer order in the process. With FM's recommendation, 7 groups are involved in this case: $A \rightarrow C \rightarrow B \rightarrow H \rightarrow I \rightarrow J \rightarrow F$. ∎

A potential drawback of the FM algorithm is that it only relies on the current state to make group transfer decisions. In some cases, such decisions may not lead to the best ticket transfer direction toward the resolver group. In Example 1, the search should explore group D rather than B after C fails to solve the ticket.

To overcome the disadvantage of the FM algorithm, we should choose the next group based on the transition probabilities from one of the past states. The intuition is that ticket transfer decision should be made based on not only the knowledge of the current group, but also the problem analysis of the past groups. Therefore, if a group transfer decision is based upon one of the past states instead of the current state, it potentially avoids the incorrect "local" decisions made by the FM algorithm. Note that the model built here is still first-order, but based on one of the past states. We name this algorithm *First-order Multiple active State search* (FMS).

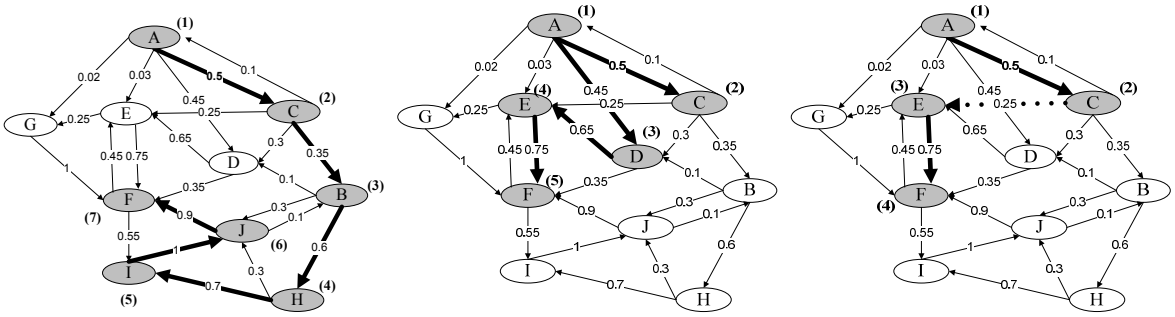
To implement this algorithm, we keep track of a *visited group set* L_v, which includes the groups visited so far, and a *candidate group set*, L_c, which consists of the unvisited neighbors of all groups in L_v. At each step, the algorithm checks all visited groups $g_{(l)} \in L_v$ and candidate groups $g \in L_c$, and selects the next group that maximizes the first-order transition probability from all $g_{(l)}$'s, i.e.,

$$g^* = \text{argmax}_g \; P(g|g_{(l)}), \forall g_{(l)} \in L_v, g \in L_c. \tag{8}$$

If g^* is not the resolver group, the algorithm will update L_v and L_c accordingly. It iterates until the resolver group is found.

Example 2: Let us revisit Example 1 using the FMS algorithm. The ticket transfer steps suggested by FMS are illustrated in Fig. 2(b). Initially, $L_v = \{A\}$ and $L_c = \{C, D, E, G\}$. The algorithm first selects group C. Now, the L_v becomes $\{A, C\}$ and L_v becomes $\{B, D, E, G\}$. Next, the algorithm selects D based on past group A, because the transition probability, $P(D|A)$, is the largest given both past groups in L_v and all candidate groups in L_c. The iteration continues until group F is selected after the algorithm visited group E. Therefore, the route suggested by FMS is $A \rightarrow C \rightarrow D \rightarrow E \rightarrow F$, which is shorter than that in Example 1. ∎

Intuitively, FMS should outperform FM, as it can avoid tracing along a wrong direction when navigating the first-order Markov model for group transfer. In the above example, it is obviously beneficial to try another direction that is more likely to be correct, i.e., group D after C fails.



(a) FM (b) FMS (c) VMS

Figure 2: Examples that compare the three ticket routing algorithms

$S_{(2)}$	next group g_i	$P\{g_i \mid S_{(2)}\}$
$\{A, C\}$	E	0.6
$\{A, C\}$	D	0.2
$\{A, C\}$	B	0.1
$\{A, C\}$	G	0.1
$\{E, C\}$	F	0.6
$\{E, C\}$	G	0.4
...	...	...

Table 3: 2nd-order transition probability: Fig. 2(c)

4.2 Variable-order Search Algorithm

Both algorithms introduced so far make ticket transfer recommendations based on 1st-order Markov model. It is possible to improve prediction accuracy further by using a higher order Markov model. Therefore, we introduce the third algorithm, called *Variable-order Multiple active State Search* (VMS).

Similar to FMS, VMS also maintains a visited set, L_v, and a candidate set, L_c. It selects a group from L_c in each iteration and expands L_v, until the resolver group is visited. Different from FMS, VMS first checks all available transfer probabilities $P(g|S_{(k)}), S_{(k)} \subseteq L_v$, for all of the groups that have been visited in the past. Then, it selects the next group g^* that maximizes the transfer probability from $S_{(k)}$,

$$g^* = \text{argmax}_g \, P(g|S_{(k)}), \forall g \in L_c, S_{(k)} \subseteq L_v. \quad (9)$$

Example 3: Let us apply the VMS algorithm to Example 1. Suppose we can use either 1st- or 2nd-order Markov model, with the 2nd-order transition probabilities listed in Table 3. The VMS algorithm works as follows. Starting from the initial $L_v = \{A\}$, since there is only one 1st-order model available at this time, the algorithm transfers the ticket to group C and updates L_v to $\{A, C\}$.

Now the algorithm has the choices of using either 1st- or 2nd-order Markov model. We find that the highest conditional probability in the 2nd-order model, $P(E|A, C) = 0.6$, is greater than that in the 1st-order model, $P(D|A) = 0.45$. So the algorithm chooses E as the next group, since the 2nd-order model predicts with higher confidence. In Fig. 2(c), we use dashed thick line to represent this transfer. From group E, $P(F|E) = 0.75$ is the highest conditional probability for all candidate groups in L_c, even compared to 2nd-order probabilities, hence F is selected next. Thus, the VMS algorithm finally reaches the resolver group F in 4 steps: $A \rightarrow C \rightarrow E \rightarrow F$. ∎

Algorithm 1 VMS(g_s)

Starting Group g_s.

```
1: Initialize L_v = {g_s}; L_c = G \ {g_s};
2: while L_c is not empty do
3:    Get g* out of L_c which satisfies Eq. (9);
4:    Add g* to L_v;
5:    if g* is the resolver group then
6:       successful, return solution g*;
7:    end if
8:    Generate Neighbor(g*) of g*;
9:    Add Neighbor(g*) - L_v to L_c;
10: end while
```

A formal description of VMS is given in Algorithm 1. The neighbor nodes of g are represented as Neighbor(g) in the algorithm. By considering more than one group visited in the past, the VMS algorithm compares all historical group transfer patterns, and finds the one with the highest confidence. Compared to FM and FMS, VMS adds more complexity to the model. It also achieves higher prediction accuracy, as verified by our experiments. The VMS algorithm is the one we recommend to use in practice.

It is worth noting that VMS may not always use a higher-order model for two reasons. First, for a new ticket, a full-fledged higher-order Markov model may not be available due to the limited size of training dataset. That is, not all transfer patterns $S_{(k)}$ were observed in the training dataset. In these cases, we reduce the order until a recognizable group transfer pattern is found. Second, in some cases a lower order conditional probability can be a stronger indicator than a higher-order one and therefore is more favorable to be used. For instance, in Fig. 2(c), if the current group is I, the 1st-order model indicates that the next group must be J (with probability 1), despite what past groups have been visited. That is why we propose VMS that considers variable orders in determining ticket transfer.

5. EXPERIMENTS

In this section, we report our experiments on evaluating the effectiveness and robustness of the proposed framework. The evaluation is based on 1.4 million problem tickets collected from IBM's current problem management system over a 1-year period from Jan 1, 2006 to Dec 31, 2006. These tickets were classified into 553 problem categories [2], e.g., `AIX`,

[2]When a ticket is first opened, the helpdesk assigns a parameter that indicates which problem domain it falls into.

DB2, Windows, etc. On average, $50 - 900$ groups (both resolver and non-resolver) were involved in solving the tickets for each problem category.

In our study, we partition the dataset into training and testing sets for each problem category. Using the training set, we first build the Markov models presented in Section 3. Then for each ticket in the testing set, given the initial group assignment, we apply the search algorithms introduced in Section 4. To evaluate the effectiveness of our system, we compare the resolution sequence of the testing tickets with the one recommended by our system. Our experiments will demonstrate:

1. Effectiveness: We show that all three search algorithms, FM, FMS, and VMS, can significantly improve ticket routing. Specifically, VMS outperforms the other two algorithms and is able to reduce the MSTR from 3.94 (based on human decisions) to 2.58 on average. We also validate our assumption that using intermediate transfer steps in model training is beneficial.
2. Robustness: We test the sensitivity of our approach, with respect to the size of training set, time-variability of tickets, and various problem categories. The results show that our solution consistently achieves good performance.
3. Case Study: We use a real ticket example to illustrate the details of how a ticket routing system can benefit from our solution.

5.1 Effectiveness

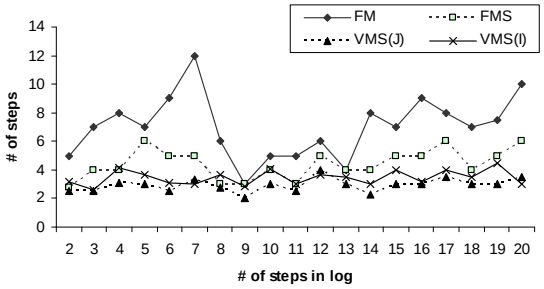


Figure 3: Effectiveness of FM, FMS, and VMS

We first describe the performance of our algorithms on a randomly chosen problem category, AIX. We derive the Markov models as discussed in Section 3 using the resolution sequences extracted from $6,695$ tickets, and then apply the FM, FMS, and VMS algorithms to the rest of $2,634$ tickets.

Fig. 3 shows the MSTR comparison between the original ticket routing and the routing recommended by our algorithms. The x-axis is the original number of transfer steps recorded in the testing dataset, while the y-axis shows the MSTR resulted from FM, FMS, and VMS, respectively. Note that all three algorithms can be applied based on either Eq. (3) or Eq. (4). Due to space limitation, we only provide the results of VMS using Eq. (3) (denoted as VMS(I)) and Eq. (4) (denoted as VMS(J)). As shown in the figure, among all three algorithms, VMS performs the best. Furthermore, VMS(J) outperforms VMS(I), which is well explained in Section 3.

Fig. 4 shows the detailed distribution (plotted in 3D)of the improvement achieved by VMS in the same experimental setting as Fig. 3. The x-axis is the number of transfer

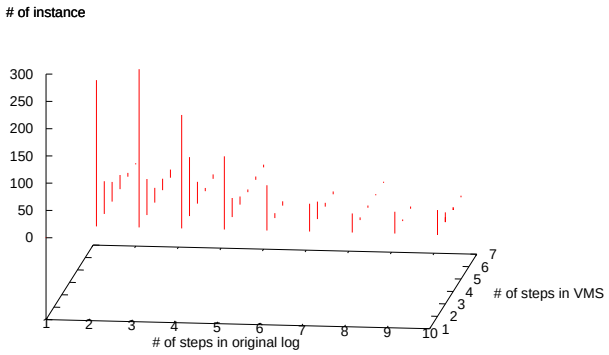


Figure 4: # of routing steps: Original vs. VMS

steps based on human decisions, the y-axis is the number of transfer steps from VMS, and z-axis is the number of tickets in different x-y combinations. As shown in the figure, VMS improves the most in those cases where human decisions led to excessively long resolution sequences (> 3). We also note that by following the recommendations from VMS, most of the tickets that originally took 2-3 steps to resolve stay the way they were, only a small portion of them follow a longer resolution sequence. In practice, these tickets are the ones that are relatively straightforward to resolve. We argue that the recommendations from VMS are unlikely to mislead human decisions in these cases. Those originally long resolution sequences represent tickets that were difficult for human experts to make correct routing decisions. It is these tickets that cause most customer dissatisfaction. The fact that VMS can significantly improve the routing efficiency in these cases makes it a good solution to routing recommendation.

Category	Original	VMS	MSTR (% off)
ADSM	5.37	3.23	37.99%
AIX	4.89	2.78	43.15%
BIOS	4.49	2.94	34.52%
DB2	4.78	2.57	46.23%
WINDOWS	3.93	2.86	27.23%
All Categories	3.94	2.58	34.52%

Table 4: MSTR for different problem categories

Table 4 shows the average MSTR for testing tickets from five categories and the average MSTR for all 553 problem categories. The fourth column is the improvement gained by VMS. It clearly indicates on average, VMS improves the transfer efficiency. As to the runtime, for each category, resolution sequence mining can be done less than 1 second, and the throughput of ticket routing is less than 1 ms per ticket, demonstrating that our solution is computationally efficient.

In Fig. 3, the fact that VMS(J) outperforms VMS(I) also suggests that our assumption that intermediate transfer steps are useful in model training is valid. We further validate this assumption as follows. We build our model using training sets that contain resolution sequences with different lengths. If our assumption is incorrect, the result based on shorter resolution sequences should be better than that based on longer ones, because shorter resolution sequences contain less intermediate transfer steps.

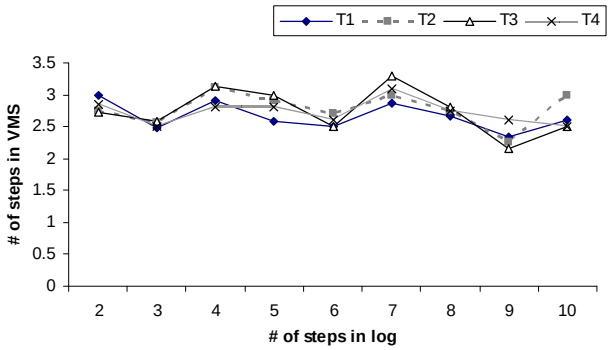


Figure 5: Consistency of the performance of VMS with different types of training sequences

Specifically, we build 4 models, T_1 through T_4, using 10 months of `AIX` tickets. We use resolution sequences of length 2 as the training set for T_1, sequences of length 3 for T_2, sequences of length 4 to 9 for T_3, and sequences of length 10 or more for T_4. The number of sequences in each training set is around 1,500. Then we use tickets closed in the next 2 months as the testing set. The effectiveness of each model is plotted in Fig. 5. It is clear that the MSTR of these models are similar, despite the fact that they were trained by resolution sequences of different lengths. This confirms that our assumption is indeed valid.

5.2 Robustness

Our next experiment aims at demonstrating the robustness of the VMS algorithm. Specifically, we test its sensitivity to a variety of factors including (1) the size of the training set, (2) time-variability of ticket data, and (3) different problem categories.

Variable Sizes of Training Set					
Training Set			Testing Set		
From	**To**	**Ticket**	**From**	**To**	**Tickets**
Jan 01	Mar 31	301,861	Apr 01	Apr 30	101,935
Jan 01	Apr 30	403,796	May 01	May 31	117,615
Jan 01	May 31	521,411	Jun 01	Jun 30	124,827
Jan 01	Jun 30	646,238	Jul 01	Jul 31	125,225
Sliding Window of Training Set					
Training Set			Testing Set		
From	**To**	**Ticket**	**From**	**To**	**Tickets**
Jan,01	Jun,30	626,238	Jul,01	Jul, 31	125,225
Feb,01	Jul,31	671,156	Aug,01	Aug, 31	132,266
Mar,01	Aug,31	602,921	Sep,01	Sep, 30	122,399

Table 5: Training and testing datasets

We first study the robustness of our approach against the size of training set. Specifically, we use 3, 4, 5, or 6-month training sets, as described in Table 5, to build the Markov models for VMS.

A comparison of MSTRs is conducted for each training set configuration. As shown in Fig. 6(a), the performance improvement resulted from VMS is consistent across all 4 training sets, while the result slightly improves as the size of the training set increases. This indicates that our algorithm is robust as long as a reasonably large training set is used. When the training set expands, our model is able to recognize more group transfer patterns. As a result, higher-order models will be available and subsequently leveraged by the VMS algorithm to improve the accuracy of predicting transfer target.

Next, we evaluate the effectiveness of our approach over time. In this evaluation, a sliding time window is applied to train the model (see Table 5), i.e., the data collected in the most recent 6 months is used as the training set and the resulting model is applied to the tickets reported in the following month. Fig. 6(b) shows that the effectiveness of our approach is also consistent over time.

The last experiment is to study the effectiveness of our approach across different problem categories. We again construct Markov models using a 6-month training set and apply it to the tickets reported in the following month. We repeated this test for 10 problem categories. Fig. 6(c) plots the detailed comparison results for 3 different problem categories: `AIX`, `DB2`, and `BIOS`. We observe that, as partially shown in Figure 6(c), the reduction in ticket transfer steps is consistent across all problem categories.

5.3 A Case Study

To better illustrate why our approach improves ticket routing, we present a case study based on a real ticket resolution sequence. The sample problem ticket is described in Table 6, which shows that 12 routing steps by 10 different expert groups were involved in resolving this ticket.

Fig. 7 depicts a fragment of the Markov model obtained from the related tickets in the training set (all in the `DB2` problem domain), with each node representing an expert group and the edge thickness representing the likelihood of transferring tickets between two groups. With VMS, only 3 groups is contacted to resolve the ticket: ⟨`SMRDX`, `SSDSISAP`, `SSSAPHWOA`⟩ (marked black in Fig. 7).

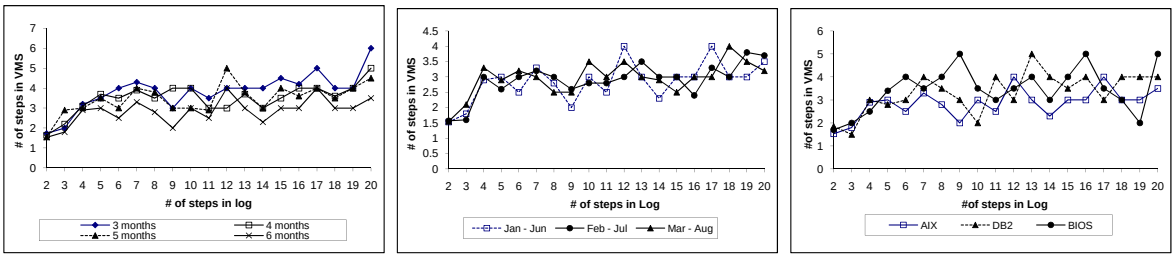
In the log, the ticket actually took 12 transfers to identify the resolver group `SSSAPHWOA` because two local decisions were made incorrectly, which transfer the ticket from `SSDSISAP` to `DRINAIXSP` (the 2nd transfer in Table 6) and from `SSMRDX` to `ASWWCUST` (the 7th transfer in Table 6). As a consequence, these wrong decisions lead to other wrong transfers (shadowed nodes in Fig. 7). Our model shows that statistically, a ticket transferred to `DRINAIXSP` will be transferred to `SSOSOGSAP` and so on if `DRINAIXSP` cannot resolve it. It seems that once transferred in a wrong direction, the ticket would not be easily routed back to correct branches.

By examining the problem status descriptions in Table 6, we find that in the process of resolving this ticket, different groups were trying to test different modules of a DB2 system, and finally came to a conclusion that DB2B needs to be recycled. What VMS recommended was that if `SMRDX` and `SDSISAP` have tried but could not solve the problem, the most efficient solution is to have `SSSAPHWOA` with higher probabilities than others to work on it and in this case, to recycle DB2B directly. This intuitively explains why our solution leads to more efficient ticket routing.

6. RELATED WORK

Process and workflow mining has been extensively studied in computer science and business research. For instance, Agrawal et al. [1] introduced an algorithm that extracts process models from event logs. Aalst et al. [20] studied the same problem in terms of Petri net. Rozinat and Aalst [17] also studied the event dependencies and applied decision trees for analyzing decision choice in business processes.

The problem of sequential pattern mining in transactions



(a) Training set size (b) Sliding time window (c) Different problem categories

Figure 6: Robustness of the VMS algorithm

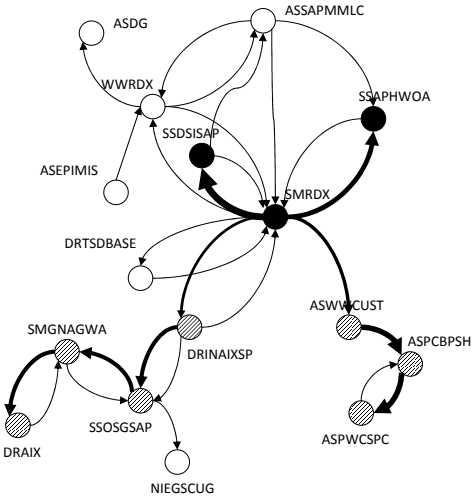


Figure 7: Markov Model (partial) in Case Study

was first illustrated by Agrawal and Srikant[2]. Various combinatorial algorithms such as SPADE [21], PREFIX [16], and SPAM [3] were developed for efficient mining in large sequence databases. In addition to combinatorial solutions, probabilistic sequence mining was also proposed [6, 7, 9, 10, 11, 18]. For instance, Cook et al. [6] tried to use neural network and Markovian approaches for mining software engineering processes. Similarly, Mannila et al. [15, 14] also used a Markovian approach to extract patterns in sequential events. A more comprehensive probabilistic model was proposed in [18]. Note that although our Markov model is conceptually similar to those of [17, 15, 14], we made significant extension to capture the data statistics and proposed a novel approach to support decision making using the derived model. Furthermore, most of previous studies were focused on discovering statistical patterns/models from event logs, rather than applying discovered models for decision making, which is the focus of this work.

In web usage mining, association rules and sequential patterns are utilized to ease users' access and improve the website design [13, 4, 19]. Kohavi et al. [12] discussed lessons and challenges from mining click stream data. In these applications, since it is difficult to tell the target web page a user was looking for from a sequence of click-through activities, the model hence built is hard to evaluate. In contrast, ticket resolution sequence does not have this issue; each sequence has a well-defined resolver group. Therefore, the model we built can be evaluated accurately. We believe resolution sequence data provides us a good platform to experiment and demonstrate the usage and the effectiveness of sequence mining.

7. CONCLUSION

In this paper, we explore the potential of mining resolution sequence to improve the efficiency of ticket routing. We develop a Markov model to statistically capture the ticket transfer decisions embedded in ticket resolution sequences. Using the developed model, we then design an algorithm to generate effective ticket routing recommendations. Through extensive experiments, we demonstrate that our approach can greatly improve the efficiency of ticket routing, especially in those cases where human experts tend to make wrong decisions and result in long resolution sequences. We also show that our model is robust to different training data sets, time-variability of tickets, and different problem categories.

The approach studied in this paper is solely based on mining ticket resolution sequence. Obviously, ticket content should also be exploited to further enhance our model. Continuing this study, we plan to extend the current model with text mining techniques, and develop a system being able to take advantage of both resolution sequence and ticket content for better routing performance.

8. REFERENCES

[1] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *Proc. Sixth Int'l Conf. Extending Database Technology*, pages 469–483, 1998.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. 1995 Int. Conf. on Data Engineering (ICDE'95)*, pages 3–14, 1995.

[3] J. Ayres, J. E. Gehrke, T. Yiu, and J. Flannick. Sequential pattern mining using a bitmap representation. In *Proc. 2002 ACM SIGKDD Int. Conf. Knowledge Discovery in Databases (KDD'02)*, pages 429 – 435, 2002.

[4] E. Han B. Mobasher, N. Jain and J. Srivastava. Web mining: Pattern discovery from world wide web transactions. In *Technical Report TR 96-050, University of Minnesota, Dept. of Computer Science*, 1996.

[5] C. Chatfield. Statistical inference regarding Markov chain models. *Appl. Statist.*, 22:7–20, 1973.

[6] J. E. Cook and A. L. Wolf. Discovering models of software processes from event-based data. *ACM Trans. Software Eng. and Methodology*, 7(3):215–249, 1998.

Entry	Description
New Ticket 29581926	GUI is failing with "Unable to Logon: RT11844: Security exception: [IBM][CLI Driver] SQL30081N A communication error has been detected. Communication protocol being used: "TCP/IP".Communication API being used: "SOCKETS". Location where the error was detected
Transferred to Group SMRDX	Contacted Mary from OSG for recycling WAS, Thanks.
Transferred to Group SSDSISAP	stopped transition on g2 and g4 and recycled WAS on e8/e9/ec/ed, then restarted transition. But still does not work.
Status updated ...	Problem Severity has been updated: Old Problem Severity: '2.' New Problem Severity:'1' The possible reason: RDC Database is unavailable to the RDC applications.
Transferred to Group DRISAIXSP	There seems to be some authorization issue, contacting Dove from SSOSGSAP for the support and further investigation
Transferred to Group SSOSGSAP	It probably database security problem, pass the ticket to SMGNAGWA
Transferred to Group SMGNAGWA	Check the DDF, which needs DRAIX support
Transferred to Group DRAIX	DDF on DB2B stopped with a mode force then restarted. Passing back to check application.
Transferred to Group SMRDX	Initial update is sent. Suggest: Ticket 29581926 transferred to Group ASWWCUST for web service checking
Transferred to Group ASWWCUST	Confirmed GUI to be up and the web services are working however, could not confirm the transition, it was not picking up so decided on to hold the transition, requested to stop the transition except WAS on KP1E8 or KP1ED.
Transferred to Group ASPCBPSH	After recycling WAS, made DDF and the DB straight which made the transition go up, Hence closing the Alert and sending MN for inputs regarding the DDF and the preventive measures for future reference
Transferred to Group ASPWCSPC	The STOP DDF was put in for change 849826, unfortunately I assumed that the 2 threads would eventually complete there by allowing DDF to stop. I should have included in the change that all remote access must have been stopped before the stop ddf command. Could not solve the problem.
Transferred to Group SMRDX	I suspect that the 2 threads that I canceled are stuck in DB2 and not rolling back so they will continue to hold the locks they have. Transfer to SSSAPHWOA for further solution
Transferred to Group SSSAPHWOA	Resolution: DB2B was recycled under 5 minutes. The problem has been solved, ticket is closed.

Table 6: Sample ticket entries

[7] J. E. Cook and A. L. Wolf. Event-based detection of concurrency. In *Proc. Sixth Int'l Symp. the Funcations of Software Eng.*, pages 35–45, 1998.

[8] L. Finesso, C.-C. Liu, and P. Narayan. The optimal error exponent for Markov order estimation. *IEEE Trans Information Theory*, 42(5):1488–1497, 1996.

[9] W. Gaaloul, S. Alaoui, K. Baïna, and C. Godart. Mining workflow patterns through event-data analysis. In *Proc. SAINT Workshops*, pages 226–229, 2005.

[10] W. Gaaloul, S. Bhiri, and C. Godart. Discovering workflow transactional behavior from event-based log. In *Proc 12th Int'l Conf. CoopIS*, pages 3–18, 2004.

[11] W. Gaaloul and C. Godart. Mining workflow recovery from event based logs. In *Proc. Third Int'l Conf. Business Process Management*, pages 169–185, 2005.

[12] R. Kohavi, L. Mason, and Z. Zheng. Lessons and challenges from mining retail e-commerce data. *Machine Learning*, 57:83–113, 2004.

[13] J. Park M. Chen and P. Yu. Data mining for path traversal patterns in a web environment. In *Proc. of the 16th Int. Conf. on Distributed Computing Systems*, pages 385–392, 1996.

[14] H. Mannila and D. Rusakov. Decomposition of event sequences into independent components. In *Proc. First SIAM Conf. Data Mining*, pages 1–17, 2001.

[15] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[16] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proc. 2001 Int. Conf. Data Engineering (ICDE'01)*, pages 215–224, 2001.

[17] A. Rozinat and W. van der Aalst. Decision mining in ProM. In *Notes in Computer Sicence*, pages 420–425, 2006.

[18] R. Silva, J. Zhang, and J. G. Shanahan. Probablistic workflow mining. In *Proc. ACM SIGKDD Int'l Conf Knowledge Discovery and Data Mining*, pages 469–483, 1998.

[19] J. Srivastava, R. Cooley, M. Deshpande, and P. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(3):12–23, 2001.

[20] W. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.

[21] M. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 40:31–60, 2001.