TAKE RISKS INTO CONSIDERATION WHILE JOB DISPATCHING

Shi-Cho Cha, Hung-Wen Tung, Chih-Hao Hsu, Han-Chao Lee, Tse-Ming Tsai and Raymund Lin
Advanced e-Commerce Institute, Institute for Information Industry, Taipei, Taiwan

Abstract: To deal with the uncertainty of job dispatching in mobile workforce management, we have proposed a framework, *Risk-Oriented joB dispAtching for mobiLe workfOrce system* (ROBALO) (Cha et al., 2005), to ease the tension between (a) the reliability requirement to serve a job request, and (b) the cost of the job's assignment. In ROBALO, the risks for workers to execute a job are taken into consideration. Such consideration is especially useful in the scenario of mobile workforce management because mobile workers usually meet unexpected situations in the field. Therefore, we can find the job assignment with the minimum cost under a certain degree of risk. Therefore, the job dispatcher can reserve enough resources and make enough preparations for a incident. Our previous work focuses on the scenario of online job dispatching, which chooses a worker or a working group to serve every incoming job request independently. In this article, we further extend to the batch model and propose a sub-optimal approach for batch job dispatching to form a complete framework.

Keywords: Mobile Workforce Management System, Online Job Dispatching, Batch Job Dispatching, Risk Management

1. INTRODUCTION

The framework of ROBALO, abbreviated term of *Risk-Oriented joB dispAtching for mobiLe workfOrce system* (Cha et al., 2005), is proposed to handle the uncertainty of job dispatching in mobile workforce management. In traditional job dispatching mechanism, exception handling processes are usually taken as the only counter-measure in dealing with the failure of job execution. Compared with this approach, ROBALO takes risks into consideration. Therefore, the time to discover the failure can be saved because we try to do things right at the first time. Furthermore, while mobile workers may usually meet the unexpected situations in the field, ROBALO would be very useful in the scenario of Mobile Workforce Management Systems (MWMSs), for it considers the uncertainty in the real world.

Simply speaking, ROBALO is made to enable a systematic approach to ease the tension between (a) the reliability required to fulfill a specific request and (b) the cost of the assignment. This mechanism would make it capable to find the assignment with the minimal cost in a certain degree of risk, so that the dilemma of reliability and cost could be balanced. Furthermore, the availability of risk information would make it possible for the job dispatcher to reserve resources and make some preparations for the exception.

In (Cha et al., 2005), it discussed the scenario of online job dispatching. Online job dispatching chooses a worker or a working group to serve every incoming job request independently, which is useful for emergency calls. When time is available, the batch model can be used to assign workers to finish several known jobs; in such case, although the cost for a specified job may increase, the overall cost may be decreased. In this article, we extend to the batch job dispatching schemes and show how ROBALO can be implemented to take risk consideration into job dispatching, so as to achieve the benefits mentioned previously.

The rest of this paper is organized as follows. Section 2 introduces the framework of ROBALO. Section 3 shows how ROBALO measures risks for a worker or a working group to do a certain job. Section 4 and Section 5 discuss how risks can be used for online and batch job dispatching respectively. Section 6 surveys related work on job dispatching in mobile workforce management. Finally, conclusions and future work are offered in Section 7.

2. OVERVIEW OF ROBALO

The architecture of ROBALO is depicted in Figure 1. First of all, the related information of job dispatching can be provided in the following:

- The *Scheduler Manager* manages the *schedules of workers*.
- The *Profile Manager* maintains the *profiles of workforce*, the *workforce's current statuses*, and the *job execution historical logs*. The profile of workforce contains workers' basic information, capacities, and other demographic information. The workforce's current statuses include workers' current location, availability information, and etc., which can be tracked by the *status monitor*. (But, it is beyond the scope of this paper to discuss status monitoring.) The results of *job execution historical logs* are recorded for a worker or a working group to execute a job.
- The *Cost Evaluator* maintains the cost for a person to execute a job in a *cost matrix* and other auxiliary information, such as the traffic cost information, to calculate the cost for a service provided by the worker.
- Finally, the *Case Manager* maintains the profiles of each case and the historical logs of job execution failure.

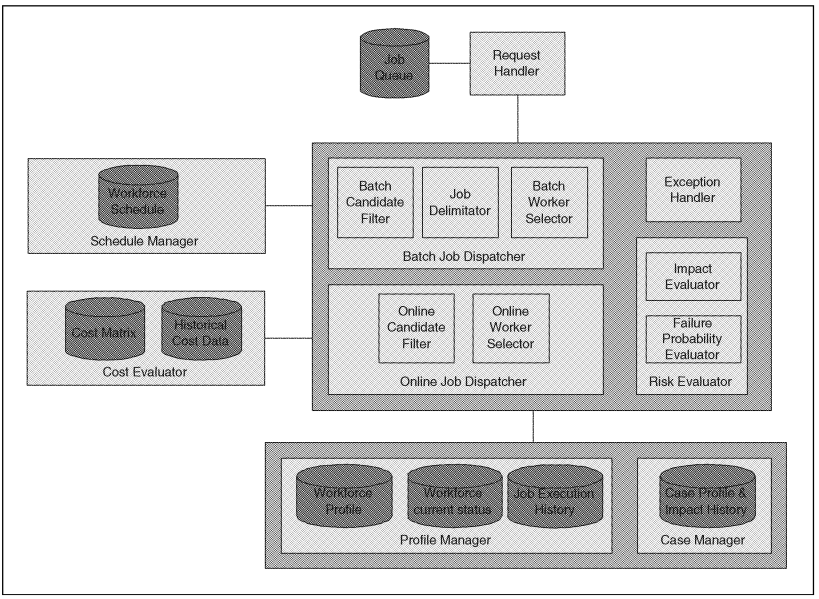

Figure 1. ROBALO Architecture

When a service request is received, the *Request Handler* parses the requests to extract the context information and generate preferences and constraints of the service. Provided that the request is required to be processed immediately, it is firstly sent to the *Online Job Dispatcher*. The *Online Candidate Filter* is in charge of filtering out the workers that are not capable of doing this work and obtains a candidate list for this job. The *Risk Evaluator* predicts the risks for a worker or working group in charge of the job in the candidate list, while the risks are calculated on the basis of the predicted loss and probability of failure, which will be further discussed in details in Section 3. Then, the *Online Worker Selector* selects the workers or working groups to serve the job request on the basis of their risks and costs. In the end, the job will be dispatched to the selected workers.

If requests do not need to be processed online, they can be kept in the *job queue*. ROBALO is used to group the requests into batches by time periods, e.g. 24 hours, and to send the various jobs to the *Batch Job Dispatcher* by batch. For example, in a job dispatching center, the jobs are assigned on the daily basis at 10:00pm for the next day. The time of a job batch can be further divided by *Job Delimitator* into several sub-periods. In each sub-period, a worker can only be assigned to do one job at this period. With such simplification, local optimization solution in each sub-period can be found. The

details are as shown in Section 5. The Batch Job Dispatcher executes its jobs dispatching sub-period by sub-period as follows: At first, similar with the Online Candidate Filter, the *Batch Candidate Filter* excludes out the workers who are not qualified to do any of the jobs in the sub-period. The Risk Evaluator predicts the risks of a job to be assigned to each worker. The Cost Evaluator estimates the cost for a worker to fulfill the task assigned including the job execution, the travel costs, and etc. The *Batch Job Selector* then tries to find out the minimized cost solution with the risks constraints.

Finally, there are few more factors to be taken into account, such as some jobs which no qualified workers could be found, in either online or batch job dispatching process, or in the case that even though the risks have been looked after, the execution of the job might still be failed. At this phase, the exceptions could be dealt with *Exception Handler*. However, the details of exception handling are beyond the scope of this article.

3. RISK PREDICTION

This section show how risks are estimated. In literatures, the definition of risk is extraordinarily broad and inconsistent. Some definitions adopt the traditional view of risks - the possibility or potential impact of unwanted occurrences. Other definitions include the uncertainties inherent in achieving optimal performance, success or failure in seizing available opportunities, and etc... (COSO, 2004). Herein we adopt the traditional definitions of risk (because it is widely used in the domain of information security (Tipton and Krause, 2004)) and calculate risks of a job assignment to be the multiple from the probability of execution failure with losses arising.

In remainder of this Section, Section 3.1 shows how the impact is anticipated. Section 3.2 demonstrates the estimation process of failure probability. In the Section 3.3 it illustrates how risks can be calculated from related losses and probabilities.

3.1 Impact Evaluation

Figure 2 shows the input and output of the Impact Evaluator. First of all, the context of a request can be defined as follows:

DEFINITION 1 (CONTEXT) *The context of a request r (we denote it as C_r) is represented by a m-ary tuple: (c_1,c_2,....., c_m). Each c_i represents an attribute or a feature of the context.*

With the definition of context, we can further define the impact history:

DEFINITION 2 (IMPACT HISTORY) *The impact history H can be defined as a set of tuple (C_i, I_i). For a past failure request R_i, C_i is the context of*

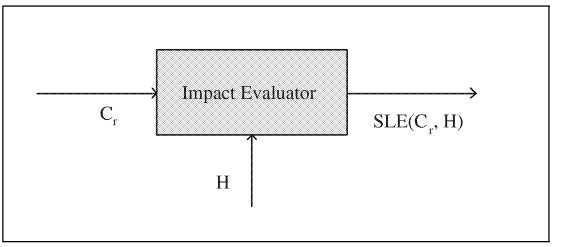

Figure 2. The I/O of the Impact Evaluator

this request. And I_i is the loss incurred from the failure execution of the job request.

The output of the Impact Evaluator is the prediction of the loss when the request r cannot be finished on time (it is denoted as $SLE(C_r, H)$). The traditional linear discriminant analysis methodology (Fisher, 1936),DK1982 can be used as follows:

- Firstly, we classify the amount of loss into several classes. For example, we can divide the amount of loss into five classes as Table 1. For each value in $Class_i$ and $Class_j$, the value in $Class_i$ is less than the value in $Class_j$ if i is less than j.
- Secondly, for each adjacent class, find the linear discriminant function $g_{i,i+1}$ with the impact history H, where $g_{i,i+1}(X) = \sum_{1 \le k \le m}(v_k \times x_k + v_0)$ and X is a m-ary tuple: (x_1,x_2,....., x_m). These linear discriminant functions are re-calculated in a certain interval so that these functions can reflect the up-to-date scenarios. For a tuple X, if $g_{i,i+1}(X) \le 0$, X belongs to $Class_k$ where $k \le i$. Otherwise, X belongs to $Class_l$ where $l > i$.
- Finally, when a context C_r is received, we calculate which class C_r belongs to as shown in Figure 3. And the ceiling of the range of the class is returned as $SLE(C_r, H)$.

3.2 Probability Evaluation

As shown in Figure 4, the Probability Evaluator predicts the probability that a working unit fails to finish a job under the context of the received request. Besides context, we have the following definitions:

DEFINITION 3 (WORKING UNIT) *A working unit W_r is the minimum set that can be assigned to serve a request r. Suppose that the universal set of workers is U ($U = \{u_i \mid u_i$ is a worker$\}$), $W_r \in U*$.*

Table 1. Classification of Impacts

Class	*Range*
$Class_1$	\$0 – \$10K
$Class_2$	\$10K – \$1M
$Class_3$	\$1M – \$100M
$Class_4$	\$100M – \$10G
$Class_5$	\$10G –

1.*m = the number of classes*
2.*for i=1, i≤m-1*
3. *if* $g_{i,i+1}(C_R) \leq 0$ *then* C_R *belongs to* $class_i$
4.*next i*
5.C_R *belongs to* $class_m$

Figure 3. The Algorithm for Context Classification.

DEFINITION 4 (JOB EXECUTION HISTORIES) *The job execution history EH can be defined as a set of tuple (C_i, W_i, S_i). For a past job request R_i, C_i is the context of this request, W_i is the work unit assigned to execute the job, and S_i is its result (either success or failure).*

DEFINITION 5 (PROFILES OF WORKFORCE) *The current profiles of workforce WP is represented by a set of x-ary tuple: (wp_{i1},wp_{i2},....., wp_{ix}). Each wp_{ij} represents an attribute or a feature of the worker u_i's profile.*

DEFINITION 6 (WORKFORCE STATUSES) *Similar to the profiles of workforce, the current statuses of workers WS is represented by a set of y-ary tuple: (ws_{i1},ws_{i2},....., ws_{iy}). Each ws_{ij} represents a kind of status of the worker u_i.*

To make it simple, we assume that the failure probabilities for each worker to execute a job is independent on another. This means that the failure probability of a work unit $P(W_r, C_r, EH, WP, WS)$ can be calculated from the product of $P(\{u_k\}, C_r, EH, WP, WS)$ where $u_k \in W_r$. Before computing $P(\{u_k\}, C_r, EH, WP, WS)$, a linear discriminant function d is found as follows:

- For each element (C_i, W_i, S_i) in EH, we extract the members of the working unit.
- For each worker u_k in the working unit W_i, we obtain the worker's profiles WP_{kt} and statuses WS_{kt} at that point of time t.

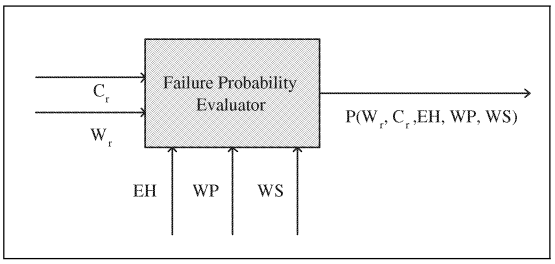

Figure 4. The I/O of the Probability Evaluator

- We concatenate WP_{kt}, WS_{kt}, and C_i into a new $(x + y + m)$-ary tuple (we call it as a working features vector (wf_{kti})). And use the tuple to find out the linear discriminant function d.
- After the linear discriminant function d is found, we calculate wf_{kti} for each working features vector and generate a set D. In the set D, we have tuples of the value of $d(wf_{kti})$ and its related result, e.g., (0.12, success), (-1, failure), (0.01, failure), etc..... And we denote it as (d_{wf_i}, s_{wf_i})

Then, $P(\{u_k\}, C_r, EH, WP, WS)$ can be calculated as follows:

- First of all, we extract the user's profiles and statuses and concatenate them with C_r into a working features vector wf_k.
- Compute the value of $d(wf_k)$.
- Compare the value of each d_{wf_i} in D and find N-nearest tuples.
- Compute the number of tuples where the value of s_{wf_i} is failure. If the number is n, the probability of failure is predicted as n/N.

Finally, $P(W_r, C_r, EH, WP, WS)$ can be obtained from the product of $P(\{u_k\}, C_r, EH, WP, WS)$ (for each $u_k \in W_r$).

3.3 Risk Calculation

DEFINITION 7 (RISK) *If we select a working group $S(S = \{u_i | u_i \, is \, a \, worker\})$ to execute a job, the risk of the job assignment R_r is the product of its loss expectancy ($SLE(C_r, H)$)) and its failure probability ($P_r(S)$).*

We assume that the failure probability for a worker to execute a job is independent to others. Therefore, we can calculate $P_r(S)$ from the product of $P(W_i, C_r, EH, WP, WS)$, for each $W_i \in S$. That is, $R_r = SLE(C_r, H) \times \prod_{W_i \in S} P(W_i, C_r, EH, WP, WS)$.

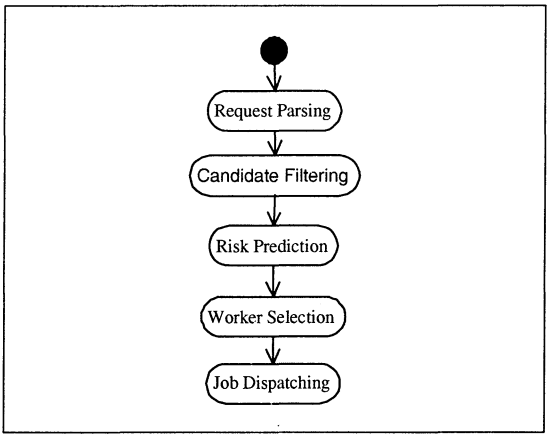

Figure 5. The Process of Online Job Dispatching in ROBALO

4. ONLINE JOB DISPATCHING

Figure 5 shows the process of online job dispatching. As described above, when a job request is received, the Request Handler parses the request to extract the context information from the request and generate constraints of the job. The Candidate Filter then uses these constraints to filter out unqualified workers and obtains a candidate list for this job.

At this point, we say a worker is capable of doing a job if he or she can satisfy the requirement of the job. For a job, its requirement includes:

- *Capability requirement.* For each job, there may be several requirement about workers' capability. For example, the worker must have a specified license or something.

- *Status requirement.* We denote the status required to execute a job J_i as SC_i (SC_i is a y-ary tuple of (sc_{i1},sc_{i2}, ...,sc_{iy})). For example, the status tuple could be a unary tuple of a location where the location is the place to execute the job. The statuses of a worker u_i is WS_{it} at time t (WS_{it} is a y-ary tuple (ws_{it1},ws_{it2},ws_{it3}, ..., ws_{ity})). In this case, a worker u_i is capable of doing a job J_j if he can change his/her status to satisfy J_j's status requirement. That is, ws_{it1} satisfies sc_{i1}, ws_{it2} satisfies sc_{i2}, etc.....

- *Time requirement.* For each job J_i, it must be executed between JS_i and JE_i. That is, a worker must be able to change its status to satisfy the status requirement of J_i before JS_i and finish the job before JE_i.

After the request context and candidate list are obtained, for each worker in the list, ROBALO evaluates his/her risk to finish the job by the product of the predicted loss and failure probability as mentioned in Section 3. And the Cost Evaluator calculates the cost for the worker to execute the job. Suppose that the cost for a worker u_i to be assigned to do a job J_j is C_{ij}. C_{ij} can be calculated from the sum of the cost (CW_{ij}) of job execution and the workers' travelling cost (CS_{ij}). The former can be obtained from a cost matrix. And the latter can be predicted by estimating the cost for a person to change his/her statuses to satisfy the status requirement before the job's time requirement JS_j.

Suppose that the maximum acceptable risk is R_a. With the candidate list L_c, the risks for a worker to execute the job, and its cost, we can select S with the pseudo-code shown in Figure 6. Generally speaking, the algorithm wishes to find S with the minimized cost where $P_r(S) \times SLE(C_r, H) \leq R_a$ or $P_r(S) \times SLE(C_r, H) \leq R_a/SLE(C_r, H)$. Obviously, if $R_a/SLE(C_r, H)$ is greater or equal to 1, any job assignment can satisfy the constraint. Therefore, we can select the worker W_i in L_c with the least cost. Otherwise, because $0 \leq P(W_i, C_r, EH, WP, WS) \leq 1$, we can apply the logarithmic function to the inequality $\Pi_{W_i \in S} P(W_i, C_r, EH, WP, WS) \leq R_a/SLE(C_r, H)$. The line 6–line 19 in Figure 6 try to find a solution to $\Sigma_{W_i \in S} log(P(W_i, C_r, EH, WP, WS)) \geq (log(R_a) - log(SLE(C_r, H)))$ with the minimized cost. The greedy algorithm is used here to choose the worker with the biggest unit cost of $log(P(W_i, C_r, EH, WP, WS))$.

5. BATCH JOB DISPATCHING

In Figure 7, it shows the process of batch job dispatching. In addition to parsing requests, differ from online job dispatching, the Request Handler collect the requests that do not need to be processed online and keeps them in its job queue. It then sends a batch of job requests required to be executed in a period of time. For a job J_i, suppose that the job is required to be executed between JS_i and JE_i, it is said that a job is required to be executed in a period of time if JS_i is within this period. Suppose that the beginning and the end of a duration is d_s and d_e, for each job J_i in d ($d_s \leq JS_i \leq d_e$), we cannot find another job J_j where $d_s \leq JS_j \leq d_e$, $JS_i < JS_j$. Based on the delimitation, we can reduce the dispatching problem to assign n single jobs to m workers.

Figure 8 gives an example about job delimitation. There are four jobs in a period P. J_0 does not belong to this period because it starts before the start time of P. We first find the job with the least termination time among the jobs started in this period. Then, we use the termination time of this job as the end of the first duration. Take Figure8 for example, at first, we find that J_1 is the job with the least termination time in P. Therefore, we can obtain the first

1. *If* $R_a/SLE(C_r, H) \geq 1$ then
2. *Select the worker* W_i *in* L_c *with the least cost*
3. S={W_i}
4. *return* S
5. *Else*
6. *current_risk = 0*
7. $W' = L_c$
8. $S = \{\}$
9. *while current_risk*>R_a *and* W' *is not empty*
10. *Find the worker* W_i *in* W' *with the biggest* $logP(W_i, C_r, EH, WP, WS)/C_{W_i}$
11. $W' = W' - W_i$
12. $S = S \cup W_i$
13. *current_risk* = $SLE(C_r, H) \times \Pi_{W_i \in S} P(W_i, C_r, EH, WP, WS)$
14. *End While*
15. *If current_risk*>R_a *and* W' *is empty then*
16. *The result cannot be found*
17. *return* ϕ
18. *End If*
19. *return* S
20. *End If*

Figure 6. The Pseudo-code for Worker Selection.

duration D_0 where the start time of D_0 is the start time of P and the end time of D_0 is the termination time of J_1. Then, we find J_3, which is the job with the least termination among the jobs in the remaining period. Therefore, the duration D_1 can be defined by using the end time of D_0 as the start time of D_1 and the termination time of J_3 as the end time of D_1. Similarly, we can further divide the remaining period into durations until there is no jobs starting at the remaining period.

After dividing the period into durations, we dispatch jobs duration by duration. To dispatch jobs in a duration, the Batch Candidate Filter filters out the workers who are not capable of doing any jobs in this duration. It is the similar case of Online Candidate Filter mentioned in Section 4 except that it takes several works into consideration at the same time.

At this point, we have a candidate list of workers and jobs needed to be dispatched in the duration. ROBALO then estimates the risk and cost for a worker in the candidate list to be assigned to do a job. We predict risk in the same way as online job dispatching. However, we need to make some modifications while estimating the cost. In online dispatching, we use a worker's current statuses to estimate the cost for the worker going to do a job by predicting the cost of the worker to change from their current statuses to the statuses satisfied with the requirement of the job. However, in batch job dispatching, a worker's statuses may not be known unless his or her previous schedule is determined. For example, if a worker is assigned to do a job in place A from

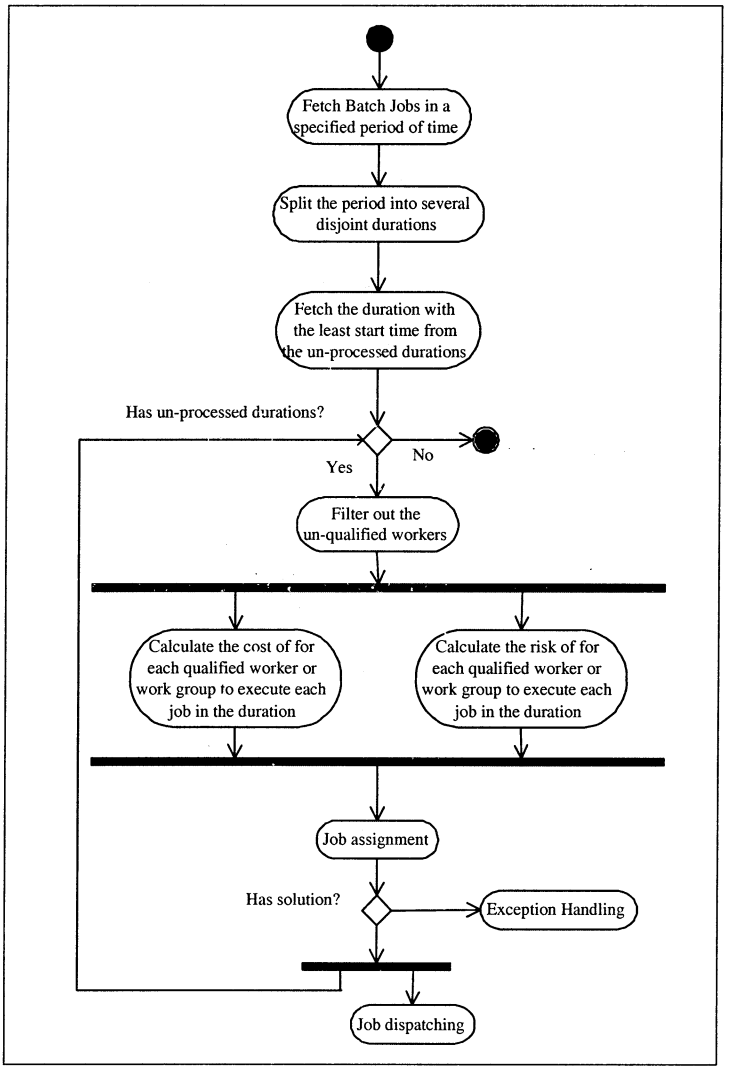

Figure 7. The Process of Batch Job Dispatching in ROBALO

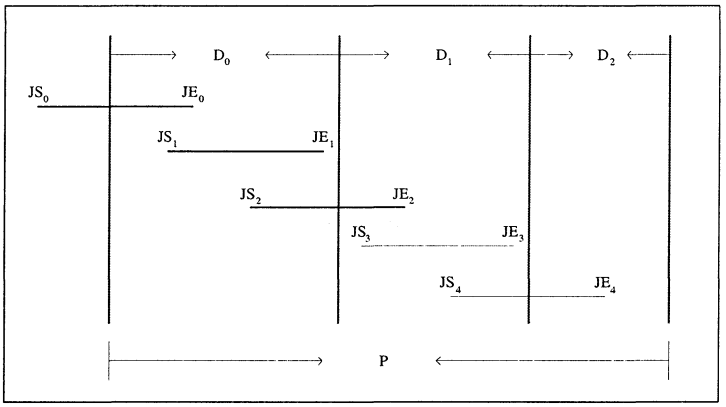

Figure 8. A Example of Job Delimitation

place B in the afternoon, the travelling cost can be reduced if the worker is assigned to do another job in place A in the morning.

Because we do job assignment from the earliest duration, when we wish to estimate the cost of changing a worker's statuses to acceptable statuses for a job, we can find the nearest previous statuses of the worker. Therefore, we can estimate the cost of changing the worker's statuses in that point to the acceptable statuses of the job and use it to calculate the cost for the worker to be assigned to do a job.

Before assigning jobs to capable workers, the Batch Worker Selector scans the risk (R_{ij}) for a worker u_i to do a job J_j. If we find that the risk is higher than a predefined acceptable level of risk R_a ($R_{ij} > R_a$), the Batch Worker Selector set related cost as infinite to present the job to be assigned to the worker.

Then, the Batch Worker Selector can simply assign workers to the jobs by solving the following linear programming problem:

Minimize $\Sigma_{u_i \in L_c, J_j \in J} C_{ij}$ (C_{ij} is the cost for worker u_i to do job J_j. L_c is the set of qualified candidates), subject to:

$\Sigma_{u_i \in L_c} x_{ij} = 1$

$x_{ij} = 1$ *or* 0 (x_{ij}=1 if J_j is assigned to u_i)

Based on the above procedure, we can make sure that we can find a local optimization solution with the minimized costs for batch job dispatching.

6. RELATED WORK

Traditional job dispatching methodologies can be classified into the following two categories: (1) the batch model and (2) the online model.

The batch model assumes that the job requests and available workers are known in advance. Then the available workers are assigned to deal with these given jobs. The batch model is useful for assigning the routine or scheduled jobs. However, in some cases, we cannot know every job requests before. For example, an emergency center may receive an emergency call from a injured person and need to dispatch an ambulance to take him/her to a nearby hospital. At this point, we can use online model to assign one or more workers to serve the incoming request.

Current online job dispatching schemes usually assign workers to serve a incoming job request based on the workers' cost, capabilities, or current statuses. For example, HAMS (Healthcare Alert Management System)(Chiu et al., 2004), (Aydin et al., 2004) classifies staffs in a hospital into different roles based on their capacities. And each kind of tasks (or alerts) can only be assigned to the staffs with appropriate roles. When a alert is triggered, HAMS select a person from the available staff members who can play the roles required for the alert and dispatches the alert to him/her. The SOS Alarm (Normark, 2002) dispatch ambulances for emergency calls based on the proximity to the ambulance stations and the status of each ambulance.

Besides workers' cost, capacities and status, we propose to take the probability that a person may fail to finish a job into consideration and use risks as a factor for job assignment in this article. While we evaluate the risk before job assignment, we can predict the expected loss of the assignment and make some "preparation" (e.g., we can assign another person as backup) while the job is dispatching. Traditional job dispatching systems usually initiate their exception handling processes after it finds that assigned workers fail to finish a job. For example, HAMS (Chiu et al., 2004) and (Aydin et al., 2004) sends messages to all staffs with the same role as it finds that a staff does not confirm a job assignment. At this point, in comparison with traditional exception handling approach, the time to discover the failure can be saved because ROBALO tries to do things right at the first time.

Moreover, risk also provides a systematic way to balance the consideration between reliability and cost consideration. This is because ROBALO assigns just enough workers so that a job can be finished within a specified level of risk. Therefore, resources can be used in a efficient way because a job dispatcher can get rid of assigning many workers to do a job blindly.

7. CONCLUSION AND FUTURE WORK

In this article, the implementation issues about ROBALO are described in details. Generally speaking, ROBALO takes the risks for a worker or a working group failing to execute a job assignment into consideration. Such consideration is especially useful in the scenario for mobile workforce management

because mobile workers may usually meet unexpected situations in the field. With ROABLO, the tension can be eased between (a) the reliability requirement to serve a job request, and (b) the cost of the job's assignment by finding the assignment with the minimum cost under a certain degree of risk. Therefore, job dispatcher can reserve enough resources and make enough preparation for an incident. In traditional job dispatching mechanism, the exception handling processes are taken to deal with the failure of job execution. In comparison with this approach, time of failure discovery can be saved because the rights things are done at the first time.

Other than a concrete implementation of ROBALO, there are many interesting things left to be done. First of all, in the current design of ROBALO, it assumes that the failure events are independent, while the relationship between workers is taken into considered. Secondly, in batch job dispatching, a period is divided into several sub-periods so as to find the local optimal solution with linear programming schemes. Moreover, it is assumed that a job can only be assigned to one worker for simplicity. It is rather a complicated however interesting problem worthy for further studies to find an overall optimal solution. Finally, although, in the ROBALO system, linear discriminant programming analysis is used to predict the risks from history data, more complicated approaches, such as non-linear programming technologies, could be included.

ACKNOWLEDGEMENTS

This research was supported by the Service Web Technology Research Project of Institute for Information Industry and sponsored by MOEA , ROC

References

Aydin, N., Marvasti, F., and Markus, H. S. (2004). Embolic doppler ultrasound signal detection using discrete wavelet transform. *IEEE Trans. on IT in Biomedicine*, 8(2):182–190.

Cha, S.-C., Tung, H.-W., Lee, H.-C., Tsa, T.-M., and Lin, R. (2005). Robalo: A risk-oriented job dispatching mechanism for workforce management system. In *IFIP I3E2005*.

Chiu, D. K. W., Kwok, B. W. C., Wong, R. L. S., Cheung, S.-C., and Kafeza, E.(2004). Alert-driven e-service management. In *HICSS*.

COSO (2004). *Enterprise Risk Management – Integrated Framework*. COSO.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annal of Eugenics*, 7:179–188.

Normark, M. (2002). Sense-making of an emergency call: possibilities and constraints of a computerized case file. In *Proceedings of NordiCHi'02*, pages 81–90, New York, NY, USA. ACM Press.

Tipton, H. F. and Krause, M. (2004). *Information security management handbook 5-th ed.* CRC Press. ISBN 0-8493-1997-8.