Acceptance Strategies for Maximizing Agent Profits in Online Scheduling

Mengxiao Wu[1], Mathijs de Weerdt[2], and Han La Poutré[1]

[1] Center for Mathematics and Computer Science (CWI), The Netherlands
[2] Delft University of Technology, The Netherlands
{wu,hlp}@cwi.nl, m.m.deweerdt@tudelft.nl

Abstract. In the global logistics market, agents need to decide upon whether to accept jobs offered sequentially. For each offer, an agent makes an immediate selection decision with little knowledge about future jobs; the goal is to maximize the profit. We study this online decision problem of acceptance of unit length jobs with time constraints, which involves online scheduling. We present theoretically optimal acceptance strategies for a fundamental case, and develop heuristic strategies in combination with an evolutionary algorithm for more general and complex cases. We show experimentally that in the fundamental case the performance of heuristic solutions is almost the same as that of theoretical solutions. In various settings, we compare the results achieved by our online solutions to those generated by the optimal offline solutions; the average-case performance ratios are about 1.1. We also analyze the impact of the ratio between the number of slots and the number of jobs on the difficulty of decisions and the performance of our solutions.

Keywords: Online decisions, Resource allocation, Admission control.

1 Introduction

Consider a market of global logistics in which a large number of jobs are dispatched day and night to many logistics companies. During a period of time, each company gets sequential offers of jobs from the market. Given its limited capacity and time resources, usually, a company can only accept part of the offers. Because of the competition in the market, we suppose the selection decisions are *immediate* and *irrevocable*. The company's target is to maximize its profit through selecting (and executing) jobs. This is an online decision problem, as the company makes the decision on each job offer without prior knowledge of future jobs. To solve it, we make an agent-based model for simulating the decision process of the company (an agent) in the market and design *acceptance strategies* (algorithms) for the agent's optimal decisions.

We first introduce our problem briefly. When job offers arrive one at a time, each job is characterized by a time window for scheduling and a payment. The agent needs to make a take-it-or-leave-it decision immediately. The agent must schedule and execute every accepted job within its time window so as to get its payment. The utility (profit) that the agent would get is the sum of the payments of all accepted jobs. In

E. David et al. (Eds.): AMEC/TADA 2011, LNBIP 119, pp. 115–128, 2013.

our analysis, we assume all jobs have the same processing time, i.e. one time slot, and the agent can execute only one job in each time slot. In this work, we focus on the selection decisions, so we make the scheduling part relatively easy, in which all jobs are assumed to be future activities and no execution happens during the whole offering process.

Our problem may be categorized as a variant of online admission control for interval scheduling [1,2,3]. In such problems, the interval between a job's release time and deadline equals the time window in our problem. The authors emphasize the immediate notification of whether to schedule each job at its arrival, which is similar to our selection decision. The decisions in our problem, however, are made at the jobs' offering time, which is not their release time, i.e. the earliest available time for execution. Hence, an accepted job can be rescheduled (within its time window) during the whole offering process. This point distinguishes our problem from almost all online interval scheduling/selection problems in previous work, in which decisions are made at the jobs' release time. Because no job will be executed during the decisions of the jobs that followed, the scheduling part of our problem is more flexible, which increases the complexity of selection. The reason is that with such flexibility in scheduling, the agent has higher expectations of future jobs, but these can also cause him to reject current jobs with good payments that he would otherwise accept.

The agent makes the decision on each job offer in two steps, i) whether this job can be feasibly scheduled together with all previously accepted jobs, and ii) when one or more feasible schedules exist, whether this job is worthy of taking. The focus of this work is on the acceptance strategies rather than the scheduling algorithms. We analyze theoretical solutions in a fundamental case and develop heuristic solutions in general and complex cases. We also present a general idea of using a theoretical analysis of a simple case to determine which are the most important parameters, and then using an evolutionary algorithm to find the optimal values of the parameters also in more complex settings. The approach presented in this work can be used to support online decisions in e-commerce applications related to logistics.

Typically, an online solution is evaluated by comparison with an optimal offline solution that knows the entire sequence of jobs in advance. In our experimental analysis, we use an *average-case performance ratio*, which is defined as the ratio between the average result generated by the optimal offline solution and the average result achieved by the online solution on a large number of instances. Our (theoretical and heuristic) solutions generate performance ratios around 1.1 in experiments with various settings. In the fundamental case, the performance of the heuristics is very close to that of the theoretically optimal online solutions. We also analyze the impact of the ratio between the number of slots and the number of jobs. The decision is most difficult when there are two to three times as many jobs as time slots.

The rest of this paper is organized as follows. We first present the problem model in Section 2 and then propose the solutions and acceptance strategies in Section 3. Following the descriptions, in Section 4, the performance of the strategies is evaluated and compared through experiments. Next, we give a brief summary of related work. Finally, conclusion and future work are given.

2 Problem Model

Suppose an agent is offered a finite set N of $n \in \mathbb{N}$ *independent* jobs sequentially. Each job $j \in N$ is characterized by a time window $[x_j, y_j]$ $(x_j, y_j \in \mathbb{N})$ and a payment $z_j \in [0, 1]$, which are independent of each other. Notice that the approach proposed by us works for any given range of payments, but we use the normalized values for ease of presentation. Every job's processing time is one time slot; it must be executed within the given time window. The agent has a set T of $t \in \mathbb{N}$ time slots available for all jobs in N. We let L denote the maximum length of all time windows where $1 \leq L \leq t$, so all jobs' time windows are in T. Given any subset of jobs $A \subseteq N$, we let $\mathcal{S}(A, T) = 1$ denote the existence of one (or more) feasible schedule such that every job $j \in A$ can be uniquely paired with a slot $i \in T$ where $x_j \leq i \leq y_j$. When a new job j is offered, the agent needs to judge whether the set of jobs $A_j \cup \{j\}$ can be feasibly scheduled first, where A_j denotes the set of jobs previously accepted before job j and $\mathcal{S}(A_j, T) = 1$. If $\mathcal{S}(A_j \cup \{j\}, T) = 1$, then the agent needs to make a decision to accept it or not, otherwise the agent can only reject it. Given the set of all accepted jobs $A \subseteq N$ (with $\mathcal{S}(A, T) = 1$), the utility U that the agent would get equals the sum of the payments of all accepted jobs, i.e. $U = \sum_{j \in A} z_j$.

3 Acceptance Strategies

A solution to the problem above is composed of two parts: a scheduling algorithm and an acceptance strategy. For each new job $j \in N$, we consider the scheduling problem of $A_j \cup \{j\}$ as a variant of the Bipartite Matching Problem. All slots T are on one side and all jobs in $A_j \cup \{j\}$ are on the other side; each job only connects to the slots of its time window. A feasible schedule is an one-sided matching in which every job is matched with one slot connected to it. We use the Ford-Fulkerson algorithm [4] to find this kind of matching between jobs in $A_j \cup \{j\}$ and slots in T. If $\mathcal{S}(A_j \cup \{j\}) = 1$, the agent then decides whether to take job j by using acceptance strategies.

We first present two theoretical strategies for a fundamental case in which all jobs have unit time windows; we analyze how to calculate the optimal values of strategy parameters, which maximize the agent's expected utility. Next, we study a general case in which the maximum length of time windows is larger than one: it is very difficult to give analytic solutions for such a setting. Therefore, we develop heuristic strategies for the general case. At last, we give extensions of our strategies for a more complex case in which the precise number of jobs is unknown.

Notice that in the rest of this paper, when we discuss the acceptance decision on a new job j, this is always based on the premise that job j can be feasibly scheduled together with previously accepted jobs in A_j.

3.1 Theoretical Strategies for Unit Time Windows

In this section, we study a fundamental case of the problem, in which every job j's time window is a single slot denoted by x_j. For theoretical analysis, we assume that the positions of all unit time windows are uniformly distributed on all slots T. We also assume that all jobs' payments are uniformly distributed on the range of $[0, 1]$.

Single Threshold. Perhaps the simplest acceptance strategy is setting a single threshold for the payments. If the new job j's payment is no less than a threshold $\alpha \in [0,1]$, the agent will accept it. We let $\mathcal{D}_j = 1$ and $\mathcal{D}_j = 0$ denote the agent's acceptance and rejection of job j respectively. The single threshold strategy is given by

$$\mathcal{D}(j) = \begin{cases} 1 & \text{if } z_j \geq \alpha \text{ and } \mathcal{S}(A_j \cup \{j\}) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

We call this the *Theoretical Single Threshold strategy* ($T1T$). Next, we present how to determine the theoretically optimal value of α, given the uniform distributions.

We let E^i denote the initially expected utility that the agent would get on each slot $i \in T$; the expected utility on all t slots is $E = t \cdot E^i$. Because t is a constant, the optimal value of α maximizing E^i also maximizes E. As we know, only if at least one job j with $(x_j = i) \wedge (z_j \geq \alpha)$ exists, slot i will finally be occupied by a job; the expected payment of the slot (and the job) is $(1+\alpha)/2$. The probability of the existence of such job j equals 1 minus the probability that no job has a time window including slot i and a payment of at least α. Reasoning in this way, E^i is given by

$$\begin{aligned} E^i =& P\left(\exists\, j,\ x_j = i \wedge z_j \geq \alpha\right) \cdot \left(\frac{1+\alpha}{2}\right) \\ =& \{1 - [P\left(j \in N,\ x_j \neq i \vee z_j < \alpha\right)]^n\} \cdot \left(\frac{1+\alpha}{2}\right) \\ =& \left[1 - \left(1 - \frac{1-\alpha}{t}\right)^n\right] \cdot \left(\frac{1+\alpha}{2}\right) \end{aligned} \tag{2}$$

We can get the optimal value of α by solving formula $\frac{dE^i}{d\alpha} = \frac{1}{2} - \frac{1}{2}\left(1 - \frac{1-\alpha}{t}\right)^n - \frac{n}{2t}(1+\alpha)\left(1 - \frac{1-\alpha}{t}\right)^{n-1} = 0$. In our experiments presented later, we solve it in an approximate way by searching α in $[0,1]$ with step size of 0.001.

n Thresholds. During the whole offering process, the agents may need to make a total of (at most) n decisions: one for each job. In this section, we present a strategy with n thresholds instead of a single threshold for all jobs. If the new job j's payment is no less than the jth threshold $\alpha_j \in [0,1]$, the agent will accept it. The strategy is given by

$$\mathcal{D}(j) = \begin{cases} 1 & \text{if } z_j \geq \alpha_j \text{ and } \mathcal{S}(A_j \cup \{j\}) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We call this the *Theoretical n Thresholds strategy* (TnT). Notice that the jth threshold α_j is independent of the jth job exactly offered.

We let E^i_j denote the expected utility that the agent would get on an available slot i when the jth job is offered. There are three possibilities. If job j's slot is slot i and its payment is no less than α_j, which happens with probability $1/t \cdot (1-\alpha_j)$, the agent will accept it and get an expected payment $(1+\alpha_j)/2$. Otherwise, if job j's payment is less than α_j (happening with probability $1/t \cdot \alpha_j$) or its slot is not slot i (happening with probability $(t-1)/t$), the agent will reject it in the expectation of slot i for the next job $j+1$. Therefore, the expected utility E^i_j is given by

Table 1. Simple example

(x_j, z_j)	$(2, 0.12)$	$(1, 0.83)$	$(3, 0.29)$	$(3, 0.41)$	$(2, 0.23)$	
$T1T$	N(< 0.295)	Y(> 0.295)	N(< 0.295)	Y(> 0.295)	N(< 0.295)	$U = 1.24$
TnT	N(< 0.435)	Y(> 0.368)	Y(> 0.282)	N($occupied$)	Y(> 0)	$U = 1.35$
Offline	N	Y	N	Y	Y	$U = 1.47$

$$E_j^i = \frac{1}{t} \cdot (1 - \alpha_j) \cdot \frac{1 + \alpha_j}{2} + \frac{1}{t} \cdot \alpha_j \cdot E_{j+1}^i + \frac{t-1}{t} \cdot E_{j+1}^i \tag{4}$$

We calculate the optimal value of α_j by solving formula $\frac{dE_j^i}{d\alpha_j} = \frac{E_{j+1}^i - \alpha_j}{t} = 0$ and get $\alpha_j = E_{j+1}^i$. So if job j's payment is no less than the agent's expectation of job $j+1$, given any available slot, the agent will accept it. Otherwise, the agent should leave the slot to job $j+1$ to get a possibly higher payment. As the expectation of job $n+1$ is zero, $\alpha_n = 0$. Replacing E_j^i and E_{j+1}^i with α_{j-1} and α_j in Eq. (4) respectively, we get a recursive function $f(j, n, t)$ to calculate threshold α_j where $1 \leq j \leq n$.

$$\alpha_j = f(j, n, t) = \begin{cases} \frac{1}{2t} \cdot (f(j+1, n, t))^2 + \frac{t-1}{t} \cdot f(j+1, n, t) + \frac{1}{2t} & j < n \\ 0 & j \geq n \end{cases} \tag{5}$$

Given a fixed t and a sequence of n, we find that i) for each setting of t and n, threshold α_j is non-linear decreasing; ii) the smaller the n is, the faster the decreasing is; iii) the smaller the n is, the lower the first threshold α_1 is. These match with the intuition that given the same number of slots, the expectations and thus the thresholds decline faster when there are less (future) jobs.

Simple Example. Given their definitions, strategy $T1T$ is theoretically optimal (in expectation) among single-threshold strategies and strategy TnT is theoretically optimal (in expectation) among n-threshold strategies in the fundamental case. We use a simple example with five jobs and three slots to illustrate the differences. In Table 1, the pairs of (x_j, z_j) in the first row represent the jobs in order of arrival (from left to right) where $1 \leq j \leq 5$. In the subsequent rows, the decisions are followed by the thresholds for both of the strategies. Although TnT loses some utility on the fourth job by accepting the third one, the advantage of the adaptive thresholds of TnT shows in accepting the last job in spite of its relatively low payment.

3.2 Heuristic Strategies

In the fundamental case above, once a job is accepted, its schedule is fixed. However, the length of time windows is generally not unit. The flexibility of (re)scheduling benefits applicability while increasing the difficulty of decisions. In this general case, even if all distributions are sill uniform, it is hard to get the optimal values of thresholds in the above way, given the multiple possibilities of time windows and tremendous possibilities of (re)scheduling. Hence, it is necessary to consider approximate solutions. In this section, we therefore develop heuristic strategies. The basic idea is using

multiple parameters to define a decision function; their optimal values are learned by an evolutionary algorithm (EA) [5] through a large number of training sessions.

Single Threshold. The first heuristic strategy proposed by us is similar to the theoretical single threshold strategy defined by Eq. (1) except that the optimal value of $\alpha \in [0, 1]$ is determined by the EA. We call this the *Heuristic Single Threshold strategy* ($H1T$); its performance is expected to be very close to that of $T1T$ in the fundamental case.

n Thresholds. Analogously, we also try a heuristic strategy of a different threshold for each job similar to the theoretical one defined by Eq. (3), but let the EA search the optimal combination of the values of those n thresholds $\alpha_j \in [0, 1]$ where $1 \leq j \leq n$. We call this the *Heuristic n Thresholds strategy* (HnT).

Three Thresholds. This strategy divides the whole offering process into three stages by using two parameters $\beta_1, \beta_2 \in [0, 1]$ ($\beta_1 < \beta_2$) and sets a single threshold $\alpha_k \in [0, 1]$ ($1 \leq k \leq 3$) for jobs' payments per stage. The agent will accept job j which is offered in the kth stage only if its payment is no less than α_k. The whole strategy is given by

$$\mathcal{D}(j) = \begin{cases} 1 & \text{if } j \leq \beta_1 \cdot n,\ z_j \geq \alpha_1, \text{ and } \mathcal{S}(A_j \cup \{j\}) = 1 \\ 1 & \text{if } \beta_1 \cdot n < j \leq \beta_2 \cdot n,\ \ z_j \geq \alpha_2, \text{ and } \mathcal{S}(A_j \cup \{j\}) = 1 \\ 1 & \text{if } j > \beta_2 \cdot n,\ z_j \geq \alpha_3, \text{ and } \mathcal{S}(A_j \cup \{j\}) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

We call this the *Heuristic* 3 *Thresholds strategy* ($H3T$).

Linear Function. To be more precise than the strategies with one or three thresholds for payments, we propose heuristic strategies based on *Piecewise Linear Functions* (PLF). As they have fewer parameters to be learned by the EA, it will be easier and faster to find the optimal solutions than the n threshold strategies. The simplest one is a linear function ($PLF1$). We set one parameter α as the slope of the linear function which generates the thresholds for payments, and also set a parameter γ to determine the constant. The agent will accept job j, if its payment is no less than the threshold given by function $p(j)$. The whole strategy is defined by

$$\mathcal{D}(j) = \begin{cases} 1 & \text{if } z_j \geq p(j) \text{ and } \mathcal{S}(A_j \cup \{j\}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where} \quad p(j) = \alpha \cdot j + \gamma \tag{7}$$

To find the global optimum of parameters α and γ for the PLF-based heuristics, we use the EA to learn these within a reasonable range. Any threshold is only reasonable within the range of $[0, 1]$, so $\gamma \in [0, 1]$. Next, given that $j \in \mathbb{N}$ and $z_j \in [0, 1]$, we can derive the range for α as follows.

$$0 \leq \alpha \cdot j + \gamma \leq 1 \text{ and } \gamma \in [0, 1] \Longrightarrow \alpha \cdot j \in [-1, 1] \Longrightarrow \alpha \in [-1, 1].$$

Two-Piece Piecewise Linear Function. The second PLF-based strategy is a two-piece piecewise linear function ($PLF2$). One parameter $\beta \in [0,1]$ cuts the whole offering process into two stages. The slopes of these two pieces are $\alpha_1, \alpha_2 \in [-1,1]$ and the constant of the first piece is $\gamma \in [0,1]$. The agent will accept job j, if its payment is no less than the threshold given by function $p(j)$. The strategy is defined by

$$\mathcal{D}(j) = \begin{cases} 1 & \text{if } z_j \geq p(j) \text{ and } \mathcal{S}(A_j \cup \{j\}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where} \quad p(j) = \begin{cases} \alpha_1 \cdot j + \gamma & \text{if } j \leq \beta \cdot n \\ \alpha_2 \cdot j + (\alpha_1 - \alpha_2) \cdot \beta \cdot n + \gamma & \text{if } j > \beta \cdot n \end{cases} \tag{8}$$

Three-Piece Piecewise Linear Function. The last one is a three-piece piecewise linear function ($PLF3$). The whole process is divided into three stages by two parameters $\beta_1, \beta_2 \in [0,1]$ where $\beta_1 < \beta_2$. The slopes of the three pieces are $\alpha_1, \alpha_2, \alpha_3 \in [-1,1]$. The constant of the first piece is $\gamma \in [0,1]$. Similarly, the thresholds are still given by function $p(j)$ and the strategy is defined by

$$\mathcal{D}(j) = \begin{cases} 1 & \text{if } z_j \geq p(j) \text{ and } \mathcal{S}(A_j \cup \{j\}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where} \quad p(j) = \begin{cases} \alpha_1 \cdot j + \gamma & \text{if } j \leq \beta_1 \cdot n \\ \alpha_2 \cdot j + (\alpha_1 - \alpha_2) \cdot \beta_1 \cdot n + \gamma & \text{if } \beta_1 \cdot n < j \leq \beta_2 \cdot n \\ \alpha_3 \cdot j + (\alpha_1 - \alpha_2) \cdot \beta_1 \cdot n & \\ \quad +(\alpha_2 - \alpha_3) \cdot \beta_2 \cdot n + \gamma & \text{if } j > \beta_2 \cdot n \end{cases} \tag{9}$$

3.3 Dealing with Uncertainty over the Number of Jobs

For the strategies presented above, the number of jobs n is required as an input. We extend the model to a more general case where the total number of jobs is unknown until the whole offering process finishes. Instead of the precise number of jobs n, the agent is only given a range of $[n_{min}, n_{max}]$ and a random distribution. In this work, we assume that n is always uniformly distributed on the range.

For the theoretical strategy $T1T$, it is straightforward to use the expected value of n to calculate the optimal value of the single threshold. This variant of $T1T$ is still theoretically optimal. However, we cannot immediately use the expected value of n in the theoretical strategy TnT, because the expected value changes after job $j > n_{min}$.

We propose an approximate solution based on TnT. We let $\bar{n}$ denote the initially expected value of n, i.e. $\bar{n} = (n_{min} + n_{max})/2$, which is consistent with the offering process until job n_{min} is offered. The agent can calculate threshold α_j by Eq. (5) with input $\bar{n}$ until $j = n_{min}$. After that $j > n_{min}$, the agent's expectation of n is changed by each new offer. We treat the distributions on the range of $[j, n_{max}]$ approximately as uniform distributions. We let $\hat{n}$ denote the average of j and n_{max}, i.e. $\hat{n} = (j + n_{max})/2$. For jobs still coming after job n_{min}, the agent calculates α_j based on $\hat{n}$ instead of $\bar{n}$. The formal definition is given by

$$\mathcal{D}(j) = \begin{cases} 1 & \text{if } z_j \geq \alpha_j \text{ and } \mathcal{S}(A_j \cup \{j\}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

where

$$\alpha_j = \begin{cases} f(j, \bar{n}, t) & \text{if } j \leq n_{min} \\ f(j, \hat{n}, t) & \text{if } n_{min} < j \leq n_{max} \end{cases}$$

$$\bar{n} = \lfloor \frac{n_{min} + n_{max}}{2} \rfloor, \ \hat{n} = \lfloor \frac{j + n_{max}}{2} \rfloor \tag{10}$$

where $f(j, n, t)$ is defined in Eq. (5).

To ensure that the heuristics define a threshold for any possible time slot, we replace n by n_{max} in their definitions. By using a representative training set for the EA, the found parameters then incorporate the distribution of n over the range of $[n_{min}, n_{max}]$.

4 Experiments

In the previous sections, we presented two theoretical strategies: $T1T$, TnT and six heuristic strategies: $H1T$, HnT, $H3T$, $PLF1$, $PLF2$ and $PLF3$. In order to evaluate and compare their performance, we set up various experiments. The experimental setting includes the number of jobs n, the number of slots t, the maximum length of time windows L, the random distribution of the starts of time windows x_j, the random distribution of the length of time windows, and the random distribution of payments z_j where $j \in N$. The length of all jobs' time windows is uniformly distributed on the range of $[1, L]$, unless the randomly generated start of the time window plus the maximum length exceeds the slots, i.e. $x_j + L - 1 > t$. In this case, the range is reduced to $[1, t - x_j + 1]$ and the length is uniformly distributed on this new range. The variable settings will be specified when we present the experiments one by one below.

Typically, the performance of online solutions is evaluated by the comparison with the problem's optimal offline solutions. The offline version of our problem is a variant of the Rectangular Assignment Problem, which can be solved by the Hungarian Algorithm [6]. In this work, we use an implementation in MATLAB [7].

Figure 1 illustrates the experimental flow that we follow for each experiment in this work. For instance, given an experimental setting, a theoretical strategy and a heuristic strategy, the experiment will be performed in two stages. First, for the heuristic strategy, use the EA to search the optimal combination of the values of its parameters, given 100 sets of n jobs. Each evaluation of the EA includes 100 simulations based on the 100 instances and the evaluation fitness is defined by the average of the 100 simulation outcomes. As the result, we get an optimal combination of the parameters' values. The heuristic strategy and the optimal values of its parameters form a heuristic solution. Repeat this part 10 times with different sets of 100 instances; 10 heuristic solutions are achieved. Second, cross-evaluate the 10 heuristic solutions by simulations with new 2000 sets of n jobs. The theoretical strategy is also evaluated on the same 2000 instances. To generate benchmarks, we also let the optimal offline solution work on the same 2000 instances in this step.

In this way, for each setting, we get 2000 results of each theoretical strategy and 10×2000 results of every heuristic strategy. We define the performance of a theoretical strategy by the average of the 2000 results. For a heuristic, the average of the 2000

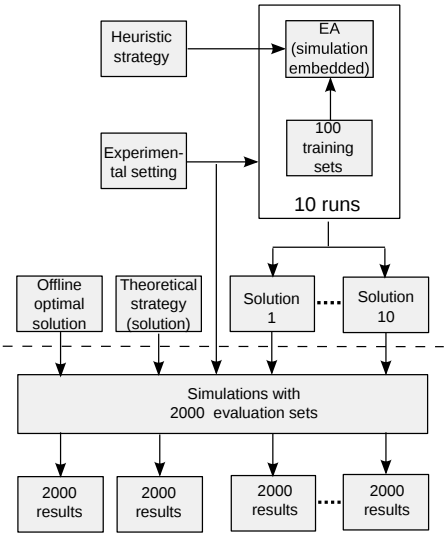

Fig. 1. Experimental flow

Table 2. Experimental settings I

t	n	t/n	L
30	$90, 75, 50, 35$	$0.33, 0.4, 0.6, 0.86$	1
$15, 50, 70$	75	$0.2, 0.67, 0.93$	1

results of each solution indicates the solution's performance. We then define the performance of a heuristic strategy by the average of the 10 averages of different solutions. We also have 2000 results of the optimal offline solution. We define the ratio between the average of the 2000 results achieved by the optimal offline solution and the performance of an online strategy to be the average-case performance ratio, which is no less than 1. The smaller the performance ratio is, the better the online solution performs.

A guideline for the EA's population size is given as at least $17 + 3 \cdot m^{1.5}$ where m is the number of parameters [5]. The number of parameters of HnT is the same as the number of jobs; the maximum n that we plan to experiment is 110. The numbers of parameters of all the other heuristics are constants: the maximum one is 6. Therefore, we set the population size as 3000 for HnT and set it as 1000 for other heuristics, which are quite sufficient. We also set the EA's evaluation limit as one million. These settings guarantee that the convergence happens before the evaluation limit is reached, so the (near) optimal results can be found.

4.1 Known Number of Jobs

First, we evaluate all strategies in two cases, unit time windows ($L = 1$) and general time windows ($L \geq 1$), under the environment that the number of jobs n is known. Both cases use the same 7 settings (Table 2); all distributions are uniform distributions.

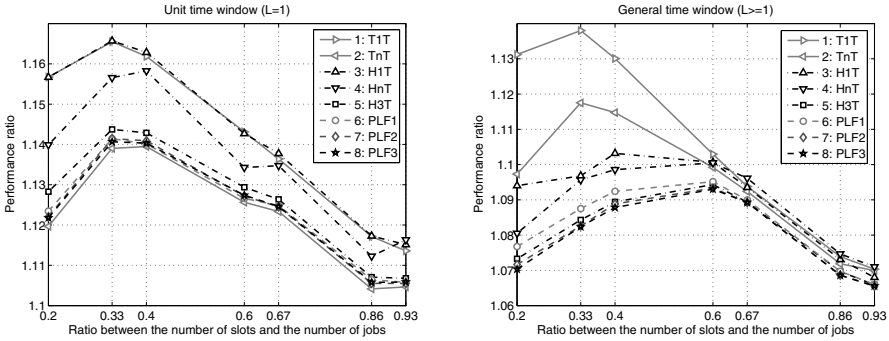

Fig. 2. Strategy performance in settings I

Unit Time Windows. We first compare these strategies in the case of unit time windows, in which strategies $T1T$ and TnT are theoretically optimal. Besides them, the n thresholds strategy is regarded as a more precise one. Therefore, we expect that strategies TnT and HnT will perform best in this case.

Figure 2 (left) illustrates the experimental results. As we expected, the performance of TnT is best in all settings. All PLF-based strategies perform very close to the benchmarks of the online solutions set by TnT; the three-piece one, i.e. $PLF3$, is the best among them. As the three thresholds of $H3T$ are constants, its performance is slightly worse than that of the PLF-based strategies. Because of the n thresholds, the performance of HnT was expected to be close to that of TnT, but it actually performs worse than the PLF-based strategies and $H3T$ here. One reason is that the size of training sets, i.e. 100 instances per evaluation, is sufficient for other heuristics but is not big enough to prevent over-fitting of the n parameters of HnT. Hence, the results are not optimal in general. By increasing the size of training sets for HnT, the problem can be resolved but the searching time will be significantly extended. The two single threshold strategies perform worst but the largest performance ratio is still small. When the single parameter of heuristic $H1T$ is learned by the EA sufficiently, its performance is almost the same as that of the theoretical strategy $T1T$.

In Figure 2 (left), we notice that the worst performance of all strategies is generated at the point of $t/n = 0.33$; the performance at its right point $t/n = 0.4$ is also low. On one side, when ratio t/n is very close to 1, as the distribution of positions is uniform, each slot is expected to assign one job. The agent's decisions are relatively easy without considering future jobs too much. On the other side, when ratio t/n is very close to 0, each slot is expected to assign many jobs. Because of the uniform distribution of payments, the decisions are also relatively easy: the agent only accepts jobs with very high payments. When the decision problem is easier, the performance of all strategies will be better. The middle area is the most difficult part, in which the agent is indeed in a dilemma between the current job and the expectation/uncertainty of future jobs. Even in this part, however, TnT and $PLF3$ can still generate performance ratios around 1.14.

Table 3. Experimental settings II

t	n	$t/\bar{n}$	L
30	$[70, 110]$, $[70, 80]$, $[35, 65]$, $[30, 40]$	0.33, 0.4, 0.6, 0.86	5
15, 50, 60	$[60, 90]$	0.2, 0.67, 0.8	5

General Time Windows. We extend to the case of general time windows. This increases the flexibility of scheduling and also the difficulty of decisions. As our theoretical strategies are derived from the case of unit time windows, their threshold values are no longer optimal in this general case. We still evaluate them here to show the change.

Figure 2 (right) illustrates the experimental results. We notice that when ratio $t/n \geq 0.6$, the performance ratios of all strategies are very close. The reason is, as we mentioned, the decision problem becomes easier in this part as the agent knows that there is a little choice on every slot. On the side of $t/n < 0.6$, the performance of strategies is clearly distinguished. Compared to $T1T$, we find that $H1T$ performs much better, although both of them use a single threshold. This indicates the advantage of the heuristic. By using the EA, the strategy can learn to find the good solutions in various settings. Compared to the results of unit time windows shown in Figure 2 (left), we find that the performance ratios are decreased (so the results are better). We will study the impact of the length of time windows on the performance of the heuristics in our future work. As we expected, the theoretical strategies (derived from the case of unit time windows) perform significantly worse than other heuristics here, because they cannot adapt to the change of the length of time windows.

4.2 Unknown Number of Jobs

The previous experiments evaluated the strategies where the number of jobs n is known. Next, we study the performance of our solutions where n is unknown but uniformly distributed over a given range . Table 3 shows a new set of 7 settings. The expectations of n in all settings correspond to the values of n in Table 2. Although strategy HnT provides good solutions in previous experiments, we omit it in the following experiments with consideration of the cost of experimental time.

Unit Time Windows. We compare the strategies under settings with unit time windows and unknown n. As we described, when we use the expectation of n instead of n for $T1T$, the resulting α is still theoretically optimal. Although the approximate variant of TnT is no longer strictly optimal in the theoretical analysis, we think the difference between the approximate solution and the theoretical solution is very small.

Figure 3 (left) illustrates the experimental results. Apparently, TnT still performs best as we expected. The value of the threshold of $T1T$ is still theoretically optimal and the performance of $H1T$ is very close to that of $T1T$. Totally, the performance ratios of all strategies in this case are very similar to those generated in the same settings (except for the issue of n) shown in Figure 2 (left). Considering the increased complexity of the problem, our solutions are robust under dynamic environments.

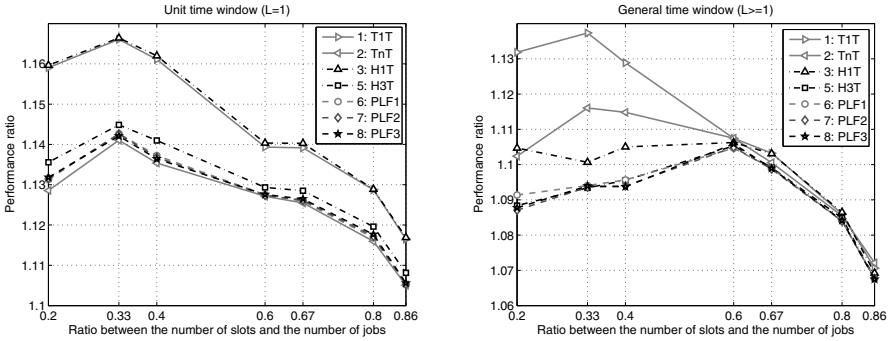

Fig. 3. Strategy performance in settings II

General Time Windows. Analogously, we also evaluate the strategies in the case of general time windows and unknown n. Figure 3 (right) illustrates the experimental results, which are quite similar to those shown in Figure 2 (right), except for the results where $t/n = 0.2$. This indicates the robust and adaptive properties of our approach of defining key parameters and using the EA to learn their optimal values.

4.3 Non-uniform Distributions

Further, we evaluate the strategies in more general and complex settings where the random distributions of the starts of time windows and the payments are non-uniform distributions. We experiment various settings, e.g. all payments being exponentially distributed or all starts being normally distributed around a slot close to one end of T; the resulting average-case performance ratios are between 1.09 to 1.22.

5 Related Work

Our problem relates to the online weighted bipartite matching problem, which is to assign each of sequentially arriving requests to one of the servers given in advance to maximize/minimize the total weight of the matching being produced [8,9]. Instead of accepting all requests, we focus on selecting a subset of requests to maximize the utility. Thus, our problem is also similar to the multiple secretary problem, which is to select the best m items out of the total $n > m$ items in an online fashion [10]. Instead of the ordinal criterion, Babaioff *et al.* present generalized secretary problems as a framework for online auctions which defines the objective in terms of the numerical values of items [11]. Different from these models, the selection problem studied by us involves a special assignment, i.e. interval scheduling [12]; this combination is also known as an online problem of admission control [1,3]. Given that jobs arrive online, a scheduler needs to choose whether to schedule each job to maximize the gain. An acceptance notification can either be given when the job really starts or be given once it can be feasibly scheduled. The latter is the same as the requirement of our problem, but our model permits all accepted jobs to be rescheduled. The scheduling part in our work

may be relatively easy, but the online acceptance decision becomes more complex. The reason is that the decision on the current job may influence the decisions about all future jobs in our problem rather than the next few jobs in the problem of interval scheduling.

The problem in [2] is more similar to our work, but the goal is different. They use greedy algorithms, e.g. accepting any job which can be feasibly scheduled (with commitment), and analyze competitive-ratios of these algorithms. We focus on the development of acceptance strategies to maximize the profit rather than the server's utilization and provide exact solutions. Their algorithm called GREEDY can indeed be used for our problem as well and is actually very similar to our single-threshold strategy with a low value. Comparing the resulting average-case performance ratios, on average our other threshold-algorithms perform much better than the GREEDY algorithm.

Summarizing, our model's uniqueness lies in the combination of scheduling and selection, which are influenced by each other during the whole decision process. Our approach also provides a new direction of solving this kind of online decision problem and we evaluate the performance of online solutions by the average-case performance ratio instead of the worst-case competitive ratio.

6 Conclusion and Future Work

In this paper, we have introduced and studied an online decision problem which requires an agent to make acceptance decisions on sequential job offers. The objective is to maximize the utility, the sum of the payments of all accepted jobs. During the whole offering process, the agent's concern is the limited time resources and the expectation of high-payment jobs in the future.

We have presented both theoretical and heuristic solutions. In a fundamental case with unit time windows and uniform distributions, when it is necessary to use the simplest one, our theoretical single threshold strategy $T1T$ can provide the optimal value of the threshold. Our theoretical n threshold strategy TnT can generate the theoretically optimal outcomes in expectation when the number of jobs n is known and still has the best performance amongst all proposed strategies when n is unknown. From fundamental settings to complex settings, compared to the optimal offline solutions, the average-case performance ratios achieved by our online solutions are around 1.1. Overall, the strategy of three-piece piecewise linear function $PLF3$ performs very close to the theoretically optimal online solution in the fundamental case and shows the best performance in all complex settings. As it only has 6 parameters determined by the EA, we say it is a high performance solution which can be specified in a short time. Other heuristics, e.g. $H1T, H3T, PLF, PLF2$, are also very good online solutions requiring even less EA searching time. Even without sufficient training, strategy HnT also generates good results and its performance can be improved if time permits.

Through the experimental analysis, we have pointed out the impact of one key factor, i.e. the ratio between the number of slots and the number of jobs t/n, on the strategy performance. When t/n is at the middle part of $[0, 1]$, the online decision is most difficult. Although the performance of our solutions is a little lower in this part, the performance ratios between 1.09 and 1.22 illustrate the advantage of our solutions for this dynamic problem. Given various settings, in which it is difficult to find any analytical clue, our solutions show their generality, robustness and adaptivity. Although we make

an assumption of unit processing time for all jobs, this work provides an approach that also applies to more generic problems involving both acceptance decisions and complex scheduling. For instance, the heuristic strategies proposed by us could be used in settings with arbitrary length jobs.

Through this work, we have learned that EAs can be used to tune the relevant parameters for settings that are hard to analyze theoretically; this thus gives a general approach, which also works for new settings (although we don't know how good it is in new settings). We answered questions such as i) how to deal with acceptance decisions and scheduling separately, ii) how to find good acceptance strategies, even if it is very hard or impossible to derive an optimal strategy (in expectation) analytically, and iii) which heuristic strategy works best (PLF3), and why (a good balance between accuracy and number of parameters).

In our future work, we would like to derive theoretically optimal solutions for general time windows in addition to our heuristic solutions. Another interesting topic is to extend the problem to a model where the processing time of jobs can vary. We may still use the approach presented in this work but need to add other key factors especially related to scheduling to achieve good results in complex environments. Analysis of competitive-ratios of our algorithms will also be included in our next work.

References

1. Goldwasser, M., Kerbikov, B.: Admission control with immediate notification. Journal of Scheduling 6(3), 269–285 (2003)
2. Garay, J., Naor, J., Yener, B., Zhao, P.: On-line admission control and packet scheduling with interleaving. In: Proc. of INFOCOM 2002, vol. 1, pp. 94–103. IEEE (2002)
3. Fung, S.P.Y.: Online Preemptive Scheduling with Immediate Decision or Notification and Penalties. In: Thai, M.T., Sahni, S. (eds.) COCOON 2010. LNCS, vol. 6196, pp. 389–398. Springer, Heidelberg (2010)
4. Ford, L., Fulkerson, D.: Maximal flow through a network. Canadian Journal of Mathematics 8(3), 399–404 (1956)
5. Bosman, P.: On empirical memory design, faster selection of bayesian factorizations and parameter-free gaussian EDAs. In: Proc. of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 389–396. ACM (2009)
6. Kuhn, H.: The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2(1-2), 83–97 (1955)
7. Buehren, M.: Algorithms for the Assignment Problem (2009), `http://www.markusbuehren.de`
8. Kalyanasundaram, B., Pruhs, K.: On-line weighted matching. In: Proc. of the Second Annual ACM-SIAM Symposium on Discrete Algorithms, p. 240. SIAM (1991)
9. Khuller Stephen, G., et al.: On-line algorithms for weighted bipartite matching and stable marriages. Theoretical Computer Science 127(2), 255–267 (1994)
10. Ajtai, M., Megiddo, N., Waarts, O.: Improved algorithms and analysis for secretary problems and generalizations. In: Proc. of the 36th Annual Symposium on Foundations of Computer Science, pp. 473–482. IEEE (1995)
11. Babaioff, M., Immorlica, N., Kempe, D., Kleinberg, R.: Online auctions and generalized secretary problems. ACM SIGecom Exchanges 7(2), 1–11 (2008)
12. Goldman, S.A., Parwatikar, J., Suri, S.: On-line Scheduling with Hard Deadlines. In: Rau-Chaplin, A., Dehne, F., Sack, J.-R., Tamassia, R. (eds.) WADS 1997. LNCS, vol. 1272, pp. 258–271. Springer, Heidelberg (1997)