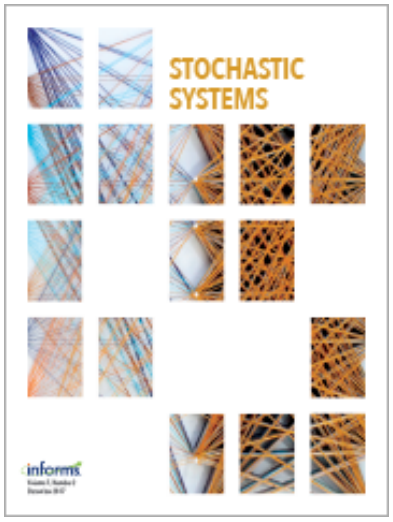
This article was downloaded by: [216.73.217.61] On: 14 July 2026, At: 07:50
Publisher: Institute for Operations Research and the Management Sciences (INFORMS)
INFORMS is located in Maryland, USA



Stochastic Systems

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Generalized Sequential Stochastic Assignment Problem

Arash Khatibi, Sheldon H. Jacobson

To cite this article:
Arash Khatibi, Sheldon H. Jacobson (2018) Generalized Sequential Stochastic Assignment Problem. Stochastic Systems 8(4):293-306. https://doi.org/10.1287/stsy.2018.0017

This work is licensed under a Creative Commons Attribution 4.0 International License. You are free to copy, distribute, transmit and adapt this work, but you must attribute this work as "*Stochastic Systems*. Copyright © 2018 The Author(s). https://doi.org/10.1287/stsy.2018.0017, used under a Creative Commons Attribution License: https://creativecommons.org/licenses/by/4.0/."

Copyright © 2018, The Author(s)

Please scroll down for article—it is on subsequent pages

With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes.
For more information on INFORMS, its publications, membership, or meetings visit http://www.informs.org

http://pubsonline.informs.org/journal/stsy/

STOCHASTIC SYSTEMS
Vol. 8, No. 4, December 2018, pp. 293–306
ISSN 1946-5238 (online)

Generalized Sequential Stochastic Assignment Problem

Arash Khatibi,[a] **Sheldon H. Jacobson**[b]

[a] Department of Industrial and Enterprise Systems Engineering, University of Illinois, Urbana, Illinois 61801; [b] Department of Computer Science, University of Illinois, Urbana, Illinois 61801

Contact: khatibi2@illinois.edu (AK); shj@illinois.edu, https://orcid.org/0000-0002-9042-8750 (SHJ)

Received: February 1, 2016
Revised: March 7, 2018
Accepted: June 12, 2018
Published Online in Articles in Advance: December 27, 2018

MSC2010 Subject Classification: 90B36, 60G99, 62M99
OR/MS Subject Classification: probability: applications; probability: stochastic model applications

https://doi.org/10.1287/stsy.2018.0017

Copyright: © 2018 The Author(s)

Abstract. The sequential stochastic assignment problem (SSAP) assigns sequentially arriving tasks with stochastic parameters (coming from a known distribution) to workers with fixed success rates so as to maximize the total expected reward. This paper studies the generalized SSAP (GSSAP), an extension of SSAP with no prior information on task values. GSSAP is described as a generalization of the secretary problem, in which the set of selected elements in the secretary problem are assigned to distinct positions in GSSAP. The weighted secretary problem is used to design two deterministic and one randomized algorithm for GSSAP. The proposed algorithms have a threshold structure: the first few stages of the problem are used as a training phase to compute thresholds. These thresholds are then used to assign tasks to workers after the training phase. These algorithms provide intuitive models to assign tasks arriving in the training phase to workers. Although the expected reward achieved by the deterministic algorithms has a better lower bound, the randomized algorithm provides fairness by assigning each task to any of the workers with equal probability.

Open Access Statement: This work is licensed under a Creative Commons Attribution 4.0 International License. You are free to copy, distribute, transmit and adapt this work, but you must attribute this work as "*Stochastic Systems*. Copyright © 2018 The Author(s). https://doi.org/10.1287/stsy.2018.0017, used under a Creative Commons Attribution License: https://creativecommons.org/licenses/by/4.0/."

Funding: This research was supported in part by the Air Force Office of Scientific Research [Grant FA9550-15-1-0100]. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Government or the Air Force Office of Scientific Research.

Keywords: online matching • secretary problem • sequential assignment

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

1. Introduction

The sequential stochastic assignment problem (SSAP) assigns sequentially arriving tasks to a set of workers so as to maximize the expected value of a reward function (Derman et al. 1972): there are n sequentially arriving tasks with random values that should be optimally assigned to n workers with fixed success rates. Associated with the jth arriving task is a random variable X_j and a value x_j, which is revealed upon its arrival. The tasks' associated random variables are assumed to be *independent and identically distributed* (*IID*). The reward of each assignment is the product of the worker's success rate and the task value assigned to the worker. The objective is to maximize the expected sum of the assignment rewards. The number of tasks and the distribution of the task's random variable are known. The main challenge is that the decision maker irrevocably assigns the arriving tasks to workers, without knowing the future task values. The optimal policy for the SSAP with n tasks computes n intervals and assigns the arriving task to the worker with the ith largest success rate if the task value is in the ith highest interval (Derman et al. 1972), as defined by Theorem 1.

Theorem 1 (Derman et al. 1972). *For each $n \geq 1$, there exists numbers*

$$-\infty = a_{0,n} \leq a_{1,n} \leq \ldots \leq a_{n,n} = +\infty,$$

such that whenever there are n workers (with $p_1 \leq p_2 \leq \ldots \leq p_n$) to be assigned to n tasks, then the optimal choice in the initial stage is to use p_i if the random variable X_1 is contained in the interval $(a_{i-1,n}, a_{i,n}]$, $i = 1, 2, \ldots, n$. The $a_{i,n}$ depend on G_X (the CDF of tasks associated random variable) but do not depend on the $\{p_i\}$. Furthermore, $a_{i,n}$ is the expected value, in an $(n-1)$ stage problem, of the quantity assigned to the i*th smallest p (assuming an optimal policy is being followed), and the total expected reward is given by*

$$\sum_{i=1}^{n} a_{i,n+1} p_i. \tag{1}$$

The constants $\{a_{i,n}\}$ are computed recursively, as described in Corollary 1. The interval thresholds depend on the number of tasks and the tasks distribution but are independent of the workers' success rates.

Corollary 1 (Derman et al. 1972). *Define* $a_{0,n} = -\infty$ *and* $a_{n,n} = +\infty$. *Then*

$$a_{i,n+1} = \int_{a_{i-1,n}}^{a_{i,n}} x\, dG_X(x) + a_{i-1,n}G_X(a_{i-1,n}) + a_{i,n}(1 - G_X(a_{i,n})), \tag{2}$$

for $i = 1,2,\ldots,n$, *where* $-\infty.0$ *and* $+\infty.0$ *are both defined to be* 0.

The secretary problem seeks to find the optimal policy for selecting the best element among a known number of sequentially arriving objects (Lindley 1961). The main challenge in the secretary problem is that the decision maker must make an irrevocable decision by comparing the relative rank of the candidates interviewed so far, without any information on the quality of the next (future) arriving candidates. Similar to the SSAP, the optimal policy of the secretary problem has a threshold structure: A number of candidates are interviewed, and the quality of the best interviewed candidate is set as a threshold. Then, the first candidate with a quality greater than the threshold is hired. Although the optimal thresholds of the SSAP are computed before the assignment process using the distribution of the task values, there is a training phase in the secretary problem to set the threshold because there is no prior information on the secretary quality values.

There are two differences between the SSAP and the secretary problem. First, whereas the secretary problem seeks to hire the best secretary, each arriving task in the SSAP is assigned to a worker with a distinct success rate value. Second, whereas one element is selected in the secretary problem, the assignment process for the SSAP begins with the arrival of the first task, and there is no training phase during which all tasks can be discarded.

The similarities between the SSAP and the secretary problem can be used to propose policies for one problem using the other problem. In one direction, the SSAP is used to derive optimal policies for the *full information* secretary problem, in which the quality values of the sequentially arriving elements are independently drawn from a known distribution. In the other direction, the secretary problem is used to generalize the SSAP by relaxing the assumption that task values come from a known distribution. The SSAP without any prior information on task values is referred to as the generalized SSAP (GSSAP).

This paper uses the weighted secretary problem (Babaioff et al. 2009) to design two deterministic policies and one randomized assignment policy for GSSAP. In the weighted secretary problem, a known number of sequentially arriving elements are used as a training phase to compute thresholds. These thresholds are then used to assign tasks to workers after the training phase. While using the same threshold structure of the algorithm for the weighted secretary problem, the proposed assignment policies for GSSAP seek to answer this question: How to assign the tasks arriving in the training phase? The main contribution of this paper is proposing three different methods to assign training phase tasks to workers and analyzing the performance by comparing the total expected reward. Although the deterministic algorithms provide slightly better lower bounds on the expected reward, the randomized algorithm is *incentive compatible*: it assigns each task to any of the workers with equal probability. This provides fairness because the sequentially arriving elements do not have an incentive to appear in specific stages to be assigned to better workers. Although the proposed algorithms start the assignment process from the first stage, they do not necessarily assign each task to one worker (i.e., some tasks might be discarded during the assignment process).

Note that the reward of an algorithm for an online matching problem without any prior information on the corresponding values of the online elements depends on the input sequence (i.e., the sequentially arriving tasks in GSSAP). Therefore, the proposed algorithms are evaluated by their competitive ratios.

Definition 1. An algorithm is α-competitive if its expected reward is at least $\frac{1}{\alpha}$ times the optimal offline reward, where the optimal offline reward is the maximum reward achieved when all elements are known in advance.

SSAP has applications in several areas, including the organ transplant, revenue management, and asset selling problem (Su and Zenios 2005, Khatibi et al. 2014) . Although assuming that task values are independently drawn from a known distribution can be used to derive the optimal policy that achieves the maximum expected reward, it is not an entirely reasonable assumption in real applications. This paper relaxes this assumption and proposes assignment policies with a guaranteed lower bound on the expected reward, which can be used in real applications with no prior information on the sequentially arriving elements. Moreover, the proposed algorithms start the assignment process from the first stage. This is particularly important in applications in which discarding tasks is very costly.

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

The remainder of this paper is organized as follows. Section 2 describes several extensions of the secretary problem and SSAP and discusses the relation between the two problems. Section 3 proposes the three assignment policies for GSSAP. Section 4 presents the results of several numerical experiments that compare the performance of the three proposed algorithms. Section 5 provides a summary of the results and concluding comments.

2. The Secretary Problem and the Sequential Stochastic Assignment

There are n sequentially arriving secretaries interviewed one by one in the secretary problem. Once a secretary is interviewed, its relative quality (i.e., the quality compared with secretaries interviewed in the previous stages) is observed. An irrevocable decision, either to hire or to reject, is made at each stage. Once rejected, a secretary cannot be recalled. On the basis of the objective and the assumptions, four different types of the secretary problem can be defined: the objective is maximizing either the probability of hiring the best secretary or the expected quality of the hired secretary. The quality values are assumed either to come independently from a known distribution (the *full information* secretary problem) or to have a random arrival order (without any assumption on the values). The random arrival order of elements is formally defined.

Definition 2. The sequentially arriving elements have a random arrival order if each element has a $\frac{1}{n}$ probability of being the jth best choice with $j = 1, 2, \ldots, n$.

The optimal policy of the secretary problem with no assumptions on the quality values and the objective to maximize the probability of hiring the best secretary divides the arriving elements into two sets. First, a set of secretaries are interviewed without anyone being hired. These secretaries are referred to as the *training set*, with a threshold defined as the quality of the best secretary in the training set. In the second phase, the first secretary that has a quality above the threshold is hired. It can be shown that as $n \to +\infty$, the optimal number of secretaries in the training phase approaches $\frac{n}{e}$ and the probability of hiring the best secretary approaches $\frac{1}{e}$ (Freeman 1983).

The optimal policy of the secretary problem with no assumptions on secretaries quality values and the objective to maximize the expected quality of the hired secretary (or equivalently, minimize the expected rank of the hired secretary) is of the following form: hire the rth secretary if their relative rank s (i.e., their rank among the secretaries interviewed so far) satisfies $s \le s^*(r)$, where $s^*(r)$ is a function of the stage number that computes the optimal relative rank. The intuition is that after a few interviews of not observing a secretary of relative rank 1, the best secretary has been rejected in the training phase with a large probability. Therefore, because the objective is to minimize the expected rank of the hired secretary (and not maximizing the probability of hiring the best secretary), a secretary of relative rank 2 is hired, and so on. Chow et al. (1964) prove that as $n \to +\infty$, the absolute rank of the secretary hired by the optimal policy tends to

$$\prod_{j=1}^{\infty}\left(\frac{j+2}{j}\right)^{1/j+1} \cong 3.8695.$$

Enns (1970) derives the optimal policy for the full information secretary problem with the objective to maximize the probability of hiring the best secretary and proves that the probability of obtaining the maximum value is independent of the distribution.

Consider the full information secretary problem with the objective to maximize the expected quality of the hired secretary $E[x_1]$, where x_1 denotes the quality value of the hired secretary. This problem is a special case of the SSAP (Derman et al. 1972): assume that n tasks with associated random values coming independently from a known distribution arrive sequentially, and there is one worker with a fixed success rate $p_1 = 1$ (and $n - 1$ workers with success rate of $p_2 = p_3 = \ldots = p_n = 0$) to be assigned one of the tasks (Khatibi and Jacobson 2014). The goal is to maximize the expected reward of the assignment, which is given by

$$E\left[\sum_{j=1}^{n} x_{i_j} p_j\right] = E[x_{i_1} p_1] = E[x_{i_1}], \tag{3}$$

where x_{i_1} denotes the task value assigned to the single available worker. This problem is equivalent to the full information secretary problem because both problems maximize the expected quality of the selected elements. The optimal policy of SSAP determines the optimal policy for the secretary problem: n optimal intervals are defined using Corollary 1, and the task is assigned to the worker with success rate $p_1 = 1$ (or equivalently the secretary is hired) if the task (secretary) value is in the highest interval $x \in (a_{n-1,n}, a_{n,n}]$ (as in Theorem 1).

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

2.1. The Multiple Choice Problem

The multiple choice secretary problem selects a subset of k secretaries so as to maximize either the expected sum of their qualities (Kleinberg 2005) or the probability of selecting the k best secretaries (Glasser and Holzager 1983). Another version of the multiple choice secretary problem seeks to maximize the probability of hiring the best secretary among the k choices (Sakaguchi 1978).

The full information multiple choice secretary problem maximizes $E[\sum_{i=1}^{k} x_i]$, where k is the number of secretaries to be hired, and x_i denotes the quality value of the ith hired secretary. The optimal policy of the SSAP can be used to maximize the expected sum of the qualities in the full information multiple choice secretary problem (Khatibi and Jacobson 2014). Assume that the workers' success rates are $p_i = 1$ for $i = 1, 2, \ldots, k$ and $p_i = 0$ for $i = k+1, k+2, \ldots, n$. Then, Theorem 1 determines the optimal policy for the SSAP that maximizes $E[\sum_{j=1}^{k} x_{i_j}]$, which is the same as the objective function of the multiple choice secretary problem. The optimal policy hires the arriving secretary if their quality value is in one of the k highest intervals defined by Corollary 1, $x \in (a_{n-k,n}, a_{n,n}]$, with x denoting the quality value, and n denoting the total number of candidates.

2.2. Other Extensions

Several other extensions of the secretary problem and the sequential assignment problem are studied in the literature. For example, Smith (1975) studies the secretary problem with uncertain employment, whereby each secretary refuses an employment offer with a fixed probability. The objective is to maximize the probability of hiring the best secretary. The doubly stochastic sequential assignment problem (DSSAP) is an extension of SSAP that can be used to study the sequential assignment problem in which a worker has a certain probability of being available for a possible assignment (Khatibi and Jacobson 2014).

Generalizing the reward function to a time-dependent model is an important extension of the basic formulations for both the secretary problem and the SSAP. Rasmussen and Pliska (1975) study the secretary problem with a discount factor α, whereas Albright (1974) models the SSAP with a time-dependent reward function. They derive the optimal policy and prove that it has a similar threshold structure.

Assuming that the number of sequentially arriving elements is limited is not realistic in several applications. Moreover, studying the problem with an infinite number of online elements provides insight on the limiting behavior of the optimal interval thresholds. Gianini and Samuels (1976) study the infinite secretary problem, whereas Albright and Derman (1972) investigate the limiting behavior of the optimal intervals as the number of assignments approaches infinity for SSAP.

Finally, the secretary problem with a random number of candidates assumes that the number of sequentially arriving candidates is random. Presman and Sonin (1972) study the problem with the objective to maximize the probability of selecting the best element. The optimal policy computes a set of (possibly) discontinuous stopping points, referred to as *islands* (i.e., whereas the optimal policy of the secretary problem hires the first relatively best candidate in stages $k, k+1, \ldots, n$, the optimal policy of the secretary problem with a random number of candidates is of the form "hire the first relatively best candidate in stages $\{k_1, k_1+1, \ldots, k_2\} \cup \{k_2+z, k_2+z+1, \ldots, k_3\} \cup \ldots$," with $z \geq 0$ a constant that depends on the distribution of the number of tasks).

Oveis Gharan and Vondrak (2011) study the secretary problem with an unknown number of candidates. They show that if the number of elements is selected by an adversary from $\{1, 2, \ldots, N\}$, then there exists a randomized algorithm that finds the best secretary with probability no less than $1/(H_{N-1}+1)$, where $H_K = \sum_{i=1}^{K} \frac{1}{i}$ is the Kth harmonic number. They further show that there is no algorithm that hires the best secretary with probability more than $1/H_N$.

Nikolaev and Jacobson (2010) consider the SSAP with a random number of tasks. They assume that the distribution of the number of tasks is known. To find the optimal policy, they define a surrogate problem as the SSAP with a fixed number of tasks N_{max}, where N_{max} is the maximum possible number of tasks. The task values in the surrogate problem (x') are computed using the distribution of the number of tasks: if $x'_{j-1} = 0$, then $x'_j = 0$. Otherwise, $x'_j = x_j$ with probability $\sum_{i=j}^{N_{max}} P_i / \sum_{i=j-1}^{N_{max}} P_i$ and $x'_j = 0$ with probability $P_{j-1} / \sum_{i=j-1}^{N_{max}} P_i$. Then, they use the SSAP with task values coming from not necessarily independent random variables (Kennedy 1986) to find the optimal policy of the surrogate problem, which is used to derive the optimal assignment of the SSAP with a random number of tasks.

3. Algorithms for the Generalized SSAP Using the Weighted Secretary Problem

This section uses the weighted secretary problem to obtain assignment policies for the GSSAP without any prior information on the task values. Babaioff et al. (2009) study the weighted secretary problem, in which up to K secretaries are selected and assigned irrevocably to K positions with weights $w_1 \geq w_2 \geq \ldots \geq w_K$ such that $\sum_{i=1}^{K} x_i w_i$ is maximized, where x_i is the value of the secretary assigned to position i, and $x_i = 0$ if position i is not filled. They

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

propose the *interval reservation algorithm*, which observes the first $l = \lfloor \frac{n}{2} \rfloor$ elements (referred to as the *training set*) and computes a set of thresholds to assign secretaries to the positions. The intervals are defined as $I_i = (\hat{x}_i, \hat{x}_{i-1})$ for $i > 1$ and let $I_1 = (\hat{x}_1, \infty)$, where $\hat{x}_i$ is the ith largest quality value in the training set. Upon arrival of a secretary with value x_e from the remaining $n - l$ secretaries (hereafter called the *selection set*), the secretary is assigned to position m with smallest index such that $m \geq m_e$, where m_e is such that $x_e \in I(m_e)$.

To analyze the interval reservation algorithm, Babaioff et al. (2009) propose an algorithm, referred to as algorithm B, which follows the same steps as the interval reservation algorithm but assigns secretary e to position m_e if the position is not yet filled. Otherwise, secretary e is not assigned to a position. Note that the expected value achieved by the interval reservation algorithm is greater than or equal to the one achieved by algorithm B because each position is filled in the interval reservation algorithm by a secretary with a value at least as large as the secretary filling the same position in algorithm B (Babaioff et al. 2009). Therefore, the competitive ratio achieved by the interval reservation algorithm is at least as good as the competitive ratio achieved by algorithm B. This result is formalized as Lemma 1.

Lemma 1 (Babaioff et al. 2009). *The expected weighted value achieved by the interval reservation algorithm is at least that achieved by algorithm B.*

Babaioff et al. (2009) prove a 4-competitive ratio for algorithm B. Then, as formalized in Theorem 2, they conclude that the interval reservation algorithm is 4-competitive.

Theorem 2 (Babaioff et al. 2009). *Algorithm B is 4-competitive. Therefore, by Lemma 1, the interval reservation algorithm is 4-competitive.*

The weighted secretary problem is equivalent to the SSAP with no prior information on task values (i.e., the GSSAP), where secretaries correspond to the sequentially arriving tasks and positions (weights) correspond to workers (success rates). The main question addressed in this section is how to assign the tasks in the training phase? Note that maximizing the number of assignments in the GSSAP is equivalent to maximizing the number of matchings in an online matching problem, which is necessary for customer satisfaction.

Sections 3.1, 3.2, and 3.3 propose three different methods to divide workers into groups recursively and assign tasks to the workers of each group using the interval reservation algorithm. Whereas the first two algorithms achieve a 4-competitive ratio, the third algorithm is a 6-competitive randomized algorithm that assigns each task to any of the workers with equal probability.

3.1. The Recursive Interval Reservation Algorithm

The *RIRA* (RIRA) recursively applies the interval reservation algorithm to groups of workers: The first group of workers includes the $l = \lfloor \frac{n}{2} \rfloor$ workers with the smallest success rates, which are assigned the tasks of the training set (i.e., the first l tasks), and the second group includes workers with the $n - l$ largest success rates, which are assigned tasks in the selection set (i.e., the last $n - l$ tasks). The tasks in the training set are assigned to the workers by recursively applying the interval reservation algorithm to the training set (i.e., workers of the first group are again divided into two groups, with the first $\lfloor \frac{l}{2} \rfloor$ tasks assigned to the group with the smallest success rates and the last $l - \lfloor \frac{l}{2} \rfloor$ tasks assigned to the group with the largest success rates, and so on). Similar to the interval reservation algorithm, the RIRA assigns the task with a value in the kth highest interval to the worker $p_{(l)}$ if l is the smallest index such that $l \geq k$. If there is no such worker, then the task is assigned to the worker with success rate $p_{(l)}$ such that l is the largest index with $l < k$. This guarantees that each task is assigned to a worker. The intuition behind the RIRA is that less skilled workers (i.e., workers with smaller success rates) are assigned tasks without a training data set (or more precisely, with a smaller training set), and more skilled workers, who have a larger effect on the total reward, are assigned tasks after a (larger) training phase.

The RIRA is formalized as Algorithm 1, with x_k denoting the kth arriving task's value and $x_{(i)}$ ($p_{(i)}$) denoting the ith largest task (success rate) value. There are $s = \lfloor \log(n) \rfloor + 1$ rounds of dividing the workers into two groups and assigning the tasks to the second group of workers using the thresholds defined by the tasks assigned to the first group. The first part of the algorithm recursively divides the workers into groups, saved in the rows of matrix B. The first group of workers A is divided into two groups in the next iteration. The second group $B(j)$, which includes the workers with the largest success rates, is assigned tasks in the $s - j + 1$th round. For example, in the GSSAP with four workers with success rates $p_{(4)} \leq p_{(3)} \leq p_{(2)} \leq p_{(1)}$, there are $s = \lfloor \log(4) \rfloor + 1 = 3$ rounds of dividing the workers. The first round divides the workers into two groups, with the first group $A_{new} = \{p_{(3)}, p_{(4)}\}$ (which is divided in the next iteration), and the second group $B(1) = \{p_{(1)}, p_{(2)}\}$ (which is assigned the last two tasks). The next round divides A_{old} (which is A_{new} of the previous iteration) into two group, with $A_{new} = \{p_{(4)}\}$ and $B(2) = \{p_{(3)}\}$. The third round includes the worker with the smallest success rate $B(3) = \{p_{(4)}\}$.

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

The second part of the algorithm assigns the sequentially arriving tasks to the groups of workers obtained in the first part. It updates the optimal intervals using the tasks assigned at each round. In the above example, $B(s) = B(3) = \{p_{(4)}\}$ includes one worker, who is assigned the first arriving task. Then, the interval thresholds are updated using the value of the first arriving task. However, because $B(2) = \{p_{(3)}\}$ includes only one worker, the second task is assigned to this worker. The third round assigns the next two tasks to the workers in $B(1)$ using the thresholds defined by the first two task values.

Lemma 2. *The RIRA achieves a reward at least as large as the interval reservation algorithm.*

Proof. We must prove that the RIRA assigns each task to a worker with a success rate at least as large as the one assigned by the interval reservation algorithm. This proves that each assignment by the RIRA achieves a reward at least as large as the interval reservation algorithm, and hence proves the lemma.

Assume that task i is assigned to the worker with success rate $p_{(l)}$ by the interval reservation algorithm. If the worker is available, the RIRA assigns task i to the same worker. This is true because the task value is placed in the same interval, and the smallest l larger than k (with k the interval where the task value is placed) is the same for both algorithms.

Assume that the worker with success rate $p_{(l)}$ is previously assigned to another task and the RIRA assigns the task to a worker with success rate $p_{(\hat{l})}$. Note that $\hat{l}$ is the largest index smaller than k (with k the interval where the task value is placed) while $l \geq k$. Therefore, $\hat{l} \leq l$ and $p_{(\hat{l})} \geq p_{(l)}$. Moreover, the number of remaining tasks and workers in each stage of the RIRA are equal. Therefore, the set of workers in the RIRA is not empty, and there is at least one $p_{(\hat{l})}$ to assign task i. □

Theorem 3. *The RIRA is 4-competitive.*

Proof. By Theorem 2, the interval reservation algorithm is 4-competitive. Lemma 2 shows that the RIRA performs at least as well as the interval reservation algorithm. Therefore, the RIRA is 4-competitive. □

Algorithm 1 (The Recursive Interval Reservation Algorithm (RIRA))

RIRA $(n, P = \{p_i | i \in \{1, 2, \ldots, n\}\})$
$A_{old} = P$
$s = \lfloor \log(n) \rfloor + 1$
for *j=1 to s* **do**
  $m = |A_{old}|$
  if $m > 1$ **then**
    $A_{new} = \{p_{(i)} \in A_{old} | i \in \{m - \lfloor m/2 \rfloor + 1, m - \lfloor m/2 \rfloor + 2, \ldots, m\}\}$
    $B(j) = A_{old} - A_{new}$
    $A_{old} = A_{new}$
  else
    $B(j) = A_{old}$
  end
end
$I_1 = (-\infty, +\infty)$
for *j=1 to s* **do**
  Assign the next r sequentially arriving tasks to workers with success rates $p_{(l)} \in B(s - j + 1)$, where $r = |B(s - j + 1)|$:
  if $x_i \in I_k$ **then**
    Assign the ith task to the worker with success rate $p_{(l)} \in B(s - j + 1)$ such that l is the smallest index with $l \geq k$; If there is no such worker, assign the task to the worker with success rate $p_{(l)} \in B(s - j + 1)$ such that l is the largest index with $l < k$
  end
  Update the intervals using the r tasks assigned:
  $I_k = (x_{(k)}, x_{(k-1)})$ for $k = 1, 2, \ldots, n + 1$, with $x_{(n+1)} = -\infty$ and $x_{(0)} = +\infty$
end

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

Lemma 3. *The reward achieved by the RIRA for the problem with n tasks and workers is at least* $\frac{1}{4}\sum_{i=n-l+1}^{n} x_{(i)}p_{(i)}$ *larger than the reward achieved by the interval reservation algorithm.*

Proof. Let $\hat{x}_i$ denote the task assigned to position i and R_{IRA} denote the total reward achieved by the interval reservation algorithm. Because the interval reservation algorithm observes the first l secretaries without assigning any of them, the total reward it achieves is given by

$$E[R_{IRA}] = \sum_{i=1}^{n} \hat{x}_i p_{(i)} = \sum_{i=1}^{n-l} \hat{x}_i p_{(i)}. \tag{4}$$

Let x_i denote the task value assigned to position i and R_{RIRA} denote the total reward achieved by the RIRA. Then,

$$E[R_{RIRA}] = \sum_{i=1}^{n} x_i p_{(i)} = \sum_{i=1}^{n-l} x_i p_{(i)} + \sum_{i=n-l+1}^{n} x_i p_{(i)} = \sum_{i=1}^{n-l} \hat{x}_i p_{(i)} + \sum_{i=n-l+1}^{n} x_i p_{(i)} = R_{IRA} + \sum_{i=n-l+1}^{n} x_i p_{(i)}. \tag{5}$$

Therefore,

$$E[R_{RIRA}] - E[R_{IRA}] \geq \sum_{i=n-l+1}^{n} x_i p_{(i)}. \tag{6}$$

Let $x^*_{(j)}$ denote the jth largest task value among the first l tasks (i.e., the training set). The first l tasks are assigned to the workers with the l smallest success rates. By 4-competitiveness of the interval reservation algorithm (Theorem 2), the reward of assigning the first l tasks is at least $\frac{1}{4}$ times the maximum offline reward of assigning the first l tasks to workers with the smallest l success rates:

$$\sum_{i=n-l+1}^{n} x_i p_{(i)} \geq \frac{1}{4} \sum_{i=1}^{l} x^*_{(i)} p_{(n-l+i)}. \tag{7}$$

Because $x^*_{(j)} \geq x_{(n-l+j)}$ for $j = 1, 2, \ldots, l$ (i.e., the best task among the first l tasks is at least as good as the $(n-l+1)$th task among all n tasks and so on),

$$\sum_{i=n-l+1}^{n} x_i p_{(i)} \geq \frac{1}{4} \sum_{i=n-l+1}^{n} x_{(i)} p_{(i)}, \tag{8}$$

which together with (6) completes the proof. □

3.2. The Deterministic Dividing Algorithm

The ith largest task value has the same probability of appearing in the first and second half of the sequentially arriving tasks in a random arrival model. Therefore, assigning all workers with larger success rates in the second half of the assignment process decreases the expected number of assignments, where the ith largest task value is assigned to the ith largest success rate (as in the optimal offline assignment). Similar to the recursive interval reservation algorithm, the *deterministic dividing algorithm* (DDA) recursively divides the workers into two groups and applies the interval reservation algorithm to each group. However, whereas the RIRA divides the workers into a group with the smallest success rates and a group with the largest success rates, the DDA divides the workers into two groups of approximately the same success rate values: the tasks in the training set, in the $n-$stage problem, are assigned to workers with success rates $\{p_{(2k)} | 2k \in \{1, 2, \ldots, n\}\}$, and the selection set's tasks are assigned to workers with success rates $\{p_{(2k+1)} | 2k+1 \in \{1, 2, \ldots, n\}\}$. This policy increases the expected number of assignments, where the ith largest task and success rate values are matched. Theorem 4 proves that the DDA, which is formally presented as Algorithm 2, is 4-competitive.

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

Algorithm 2 (The Deterministic Dividing Algorithm)

DDA $(n, P = \{p_i | i \in \{1, 2, \ldots, n\}\})$
$A_{old} = P$
$s = \lfloor \log(n) \rfloor + 1$
for *j=1 to s* **do**
  $m = |A_{old}|$
  if $m > 1$ **then**
    $A_{new} = \{p_{(i)} \in A_{old} | i = 2k\}$
    $B(j) = A_{old} - A_{new}$
    $A_{old} = A_{new}$
  else
    $B(j) = A_{old}$
  end
end
Follow the assignment process as RIRA

Theorem 4. *The DDA is 4-competitive.*

Proof. Let $p_{(n)} \le p_{(n-1)} \le \ldots \le p_{(1)}$ denote the workers success rates and $x_{(n)} \le x_{(n-1)} \le \ldots \le x_{(1)}$ denote the task values. The DDA assigns a task to the worker with the largest success rate $p_{(1)}$ exactly in the same manner as the interval reservation algorithm. Therefore, in expectation, the reward of assigning a task to this worker by the DDA is at least $\frac{1}{4}x_{(1)}p_{(1)}$.

The same procedure as in the proof of 4-competitiveness for the interval reservation algorithm is used to prove that the expected reward that the DDA achieves by assigning a task to each of the other workers is at least $\frac{1}{4}$ times the maximum offline reward. Define algorithm B in the same way as discussed in Lemma 1: if $x_j \in I_k$, algorithm B assigns the task to worker with success rate $p_{(k)}$ if and only if the worker has not already been assigned, and discards the task otherwise (Babaioff et al. 2009). It will be proven that algorithm B achieves at least $\frac{1}{4}$ of the optimal offline reward. Together with Lemma 1, this result proves that the expected reward achieved by the DDA is at least $\frac{1}{4}$ of the optimal offline reward. The DDA assigns a task to the worker with success rate $p_{(2)}$ in one of stages $z = \lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor + 1, \lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor + 2, \ldots, \lfloor \frac{n}{2} \rfloor$. Given that the task $x_{(2)}$ or a task with a larger value (i.e., $x_{(1)}$) appears in a random order at position z, then with probability $\frac{\lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor}{z-1}$ the next most valuable task (i.e., the task with the next largest value) that appeared before $x_{(2)}$ (or $x_{(1)}$) appears in the training set (i.e., in one of the stages $1, 2, \ldots, \lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor$). Conditioning on this event, with probability $\frac{\lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor - 1}{z-2}$ the next less valuable task (i.e., the task with the next smallest value) that appeared before $x_{(2)}$ (or $x_{(1)}$) appears in the training set. These two events are sufficient conditions that a task with a value at least as large as $x_{(2)}$ is assigned to the worker with success rate $p_{(2)}$ by algorithm B (Babaioff et al. 2009). The probability that the task in position z has a value at least as large as $x_{(2)}$ is $\frac{2}{n}$ (because this is equal to the probability that the task has the value $x_{(1)}$ or $x_{(2)}$). Therefore, the probability that the worker with success rate $p_{(2)}$ is assigned to a task with a value at least as large as $x_{(2)}$ by algorithm B is given by

$$\sum_{z=\lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor + 1}^{\lfloor \frac{n}{2} \rfloor} \frac{2}{n} \times \frac{\lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor (\lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor - 1)}{(z-1)(z-2)} = \frac{2}{n} \times \lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor (\lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor - 1) \times \sum_{z=\lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor + 1}^{\lfloor \frac{n}{2} \rfloor} \left(\frac{1}{z-2} - \frac{1}{z-1} \right)$$
$$= \frac{2}{n} \times \lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor \left(\lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor - 1 \right) \times \left(\frac{1}{\lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor - 1} - \frac{1}{\lfloor \frac{n}{2} \rfloor - 1} \right) = \frac{2}{n} \times \lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor \times \frac{\lfloor \frac{n}{2} \rfloor - \lfloor \frac{1}{2}\lfloor \frac{n}{2} \rfloor \rfloor}{\lfloor \frac{n}{2} \rfloor - 1}. \tag{9}$$

As $n \to +\infty$, (9) is at least $\frac{1}{4}$. Therefore, in expectation, the reward achieved by the worker with success rate $p_{(2)}$ using the DDA is at least $\frac{1}{4}$ times the reward achieved by the same worker in the optimal assignment.

To prove that the same competitive ratio holds for the reward achieved by the other workers, note that $x_{(i)}p_{(i)} \ge x_{(i+1)}p_{(i+1)}$ for $i = 3, 5, 7, \ldots$. Therefore, the reward achieved by the workers with success rates $p_{(3)}, p_{(5)}, p_{(7)}, \ldots$, is at least half of the remaining reward and hence, proving a 2-competitive ratio for the reward achieved by these workers is equivalent to proving a 4-competitive ratio for the total reward achieved by the workers with success rates $p_{(3)}, p_{(4)}, p_{(5)}, \ldots, p_{(n)}$.

Consider the worker with success rate $p_{(k)}$ with $\{k = 2l + 1 | 2l + 1 \in \{3, \ldots, n\}\}$. The probability that the task in position z has a value at least as large as $x_{(k)}$ is at least $\frac{2}{n}$ (because this is equal to the probability that the task has a value $x_{(k)}$ or $x_{(k+1)}$). Therefore, following the same steps as (9), for stages $z = \lfloor \frac{n}{2} \rfloor + 1, \lfloor \frac{n}{2} \rfloor + 2, \ldots, n$, the probability

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

that the worker with success rate $p_{(k)}$ $(k = 3, 5, 7, \ldots)$ is assigned to a task with a value at least as large as $x_{(k)}$ is given by

$$\frac{2}{n} \times \lfloor \frac{n}{2} \rfloor \times \frac{n - \lfloor \frac{n}{2} \rfloor}{n-1},$$

which is at least $\frac{1}{2}$ as $n \to +\infty$. Therefore, in expectation, the reward achieved by the workers with success rate $p_{(3)}, p_{(4)}, p_{(5)}, \ldots, p_{(n)}$ using the DDA is at least $\frac{1}{4}$ times the reward achieved by the same workers in the optimal assignment. □

Lemma 4 compares the expected reward achieved by the DDA with the minimum expected reward achieved by the interval reservation algorithm. Together with Lemma 3, this result can be used to compare the lower bound of the reward achieved by the DDA and the RIRA.

Lemma 4. *Let $E[R_{DDA}]$ denote the expected reward achieved by the DDA. Then,*

$$E[R_{DDA}] - \frac{1}{4}\sum_{j=1}^{n} x_{(j)}p_{(j)} \geq \frac{1}{4} \sum_{\{i=2k+1 | k \geq 1, 2k+1 \in \{1,2,\ldots,n\}\}} (x_{(\frac{i+1}{2})} - x_{(i)})p_{(i)} + \frac{1}{4}\sum_{i=n-l+1}^{n} x_{(i)}p_{(n)}, \tag{10}$$

where $\frac{1}{4}\sum_{j=1}^{n} x_{(j)}p_{(j)}$ is the lower bound of the expected reward achieved by the interval reservation algorithm.

Proof. Let x_{i_j} denote the task assigned to the worker with success rate $p_{(j)}$, $j = 1, 2, \ldots, n$, by the DDA. Then, the expected reward achieved by the DDA is given by

$$E[R_{DDA}] = \sum_{j} x_{i_j}p_{(j)} = \sum_{\{j=2k+1 | 2k+1 \in \{1,2,\ldots,n\}\}} x_{i_j}p_{(j)} + \sum_{\{j=2k | 2k \in \{1,2,\ldots,n\}\}} x_{i_j}p_{(j)}. \tag{11}$$

A lower bound for each of the two terms in the right hand-side of (11) is proven, which are used to prove (10). By Theorem 4,

$$\sum_{\{j=2k | 2k \in \{1,2,\ldots,n\}\}} x_{i_j}p_{(j)} \geq \frac{1}{4} \sum_{\{j=2k | 2k \in \{1,2,\ldots,n\}\}} x_{(j)}p_{(j)}. \tag{12}$$

The DDA uses the interval reservation algorithm to assign tasks to workers with success rates $p_{(1)}, p_{(3)}, \ldots, p_{(n-1)}$. In the SSAP with n tasks and workers, the interval reservation algorithm does not assign any tasks to the workers with the smallest $\lfloor \frac{n}{2} \rfloor$ success rates. To prove a tighter lower bound on the reward that the DDA achieves by assigning tasks to the workers with success rates $p_{(1)}, p_{(3)}, \ldots, p_{(n-1)}$ (i.e., the first term in the right hand-side of (11)), define $n - (n - \lfloor \frac{n}{2} \rfloor)$ auxiliary workers with success rates $p_{(n)}$, which increases the set of available workers to n workers with success rates

$$p_{(1)}, p_{(3)}, \ldots, p_{(n-1)}, p_{(n)}, p_{(n)}, \ldots, p_{(n)},$$

where the smallest $\lfloor \frac{n}{2} \rfloor$ success rates are all equal to $p_{(n)}$. By Theorem 2,

$$\sum_{\{j=2k+1 | 2k+1 \in \{1,2,\ldots,n\}\}} x_{i_j}p_{(j)} \geq \frac{1}{4}\sum_{j=1}^{n-l} x_{(j)}p_{(2j-1)} + \frac{1}{4}\sum_{i=n-l+1}^{n} x_{(i)}p_{(n)}. \tag{13}$$

Inserting (12) and (13) into (11),

$$\begin{aligned} E[R_{DDA}] = \sum_{j} x_{i_j}p_{(j)} &\geq \frac{1}{4} \sum_{\{j=2k | 2k \in \{1,2,\ldots,n\}\}} x_{(j)}p_{(j)} + \frac{1}{4}\sum_{j=1}^{n-l} x_{(j)}p_{(2j-1)} + \frac{1}{4}\sum_{i=n-l+1}^{n} x_{(i)}p_{(n)} \\ &= \frac{1}{4}\sum_{j=1}^{n} x_{(j)}p_{(j)} + \frac{1}{4} \sum_{\{i=2k+1 | 2k+1 \in \{1,2,\ldots,n\}\}} (x_{(\frac{i+1}{2})} - x_{(i)})p_{(i)} + \frac{1}{4}\sum_{i=n-l+1}^{n} x_{(i)}p_{(n)}, \end{aligned} \tag{14}$$

which completes the proof. □

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

3.3. The Random Dividing Algorithm

This section describes an incentive compatible algorithm for GSSAP. First, we define incentive compatibility for the sequential assignment problem.

Definition 3. An algorithm for the sequential assignment problem is incentive compatible if it assigns each task to each of the workers with equal probability.

Note that in the n-stage problem (with n tasks and n workers), an incentive compatible algorithm assigns each task to each worker with probability $1/n$. Incentive compatible mechanisms have been proposed for various online matching problems in recent years. For example, Buchbinder et al. (2013) study the incentive compatible secretary problem via linear programming. Goel et al. (2014) design an incentive compatible mechanism for allocating tasks to workers in the online crowdsourcing markets. Greshkov and Moldovanu (2010) study the assignment of heterogenous objects to impatient agents with privately known characteristics who arrive sequentially according to a Poisson or renewal process.

The *random dividing algorithm* (RDA) randomly orders the workers such that the ith $(i = 1, 2, \ldots, n)$ largest success rate has $\frac{1}{n}$ probability of being in the jth $(j = 1, 2, \ldots, n)$ position. It then uses the task values to compute interval thresholds. Tasks are assigned to workers according to algorithm B, described in the beginning of this section. Algorithm 3 formalizes this procedure. Whereas the RIRA and the DDA divide the workers into groups deterministically, the RDA generates groups of workers randomly. First, it is proven that the RDA is incentive compatible. Then, it is proven that it is 6-competitive.

Theorem 5. *The RDA is incentive compatible.*

Proof. Let $P^* = \{p_1^*, p_2^*, \ldots, p_n^*\}$ denote the success rates of randomly ordered workers. Induction is used to prove that each of the tasks arriving in stages $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, n$ is assigned to any of the workers with success rates $P_1^* = \{p_{\lfloor n/2 \rfloor+1}^*, p_{\lfloor n/2 \rfloor+2}^*, \ldots, p_n^*\}$ with equal probability. Because this process is recursively applied and each worker has equal probability of being in the set P_1^*, this proves that each task is assigned to each worker with equal probability. The task arriving at stage $\lfloor n/2 \rfloor + 1$ is assigned to the jth worker if the task value is in the jth interval defined by the first $\lfloor n/2 \rfloor$ tasks. Because the tasks have a random arrival order, the task value is equally likely to be in any of these intervals. Therefore, the task arriving at stage $\lfloor n/2 \rfloor + 1$ is assigned to any of the workers with success rates $\{p_{\lfloor n/2 \rfloor+1}^*, p_{\lfloor n/2 \rfloor+2}^*, \ldots, p_n^*\}$ with equal probability. As the induction assumption, assume that tasks arriving at stages $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, t-1$ are assigned to any of the workers with success rates $\{p_{\lfloor n/2 \rfloor+1}^*, p_{\lfloor n/2 \rfloor+2}^*, \ldots, p_n^*\}$ with equal probability. The probability that task t is assigned to worker j is given by

$$\Pr(\textit{task } t \textit{ is assigned to worker } j | \textit{worker } j \textit{ is available at stage } t) \times \Pr(\textit{worker } j \textit{ is available at stage } t) \tag{15}$$

By the induction assumption, because each worker has equal probability of being in the set P_1^* and each task in stages $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, t-1$ is equally likely to be assigned to any of the workers in the set P_1^*, $\Pr(\textit{worker } j \textit{ is available at stage } t)$ is equal for all workers. Using the random arrival order of the tasks, task t is assigned to each worker with equal probability. □

Lemma 5 proves the lower bound of the reward achieved by the RDA by assigning the last $n - \lfloor n/2 \rfloor$ arriving tasks to workers. Lemma 6 proves that the maximum reward of the first $\lfloor n/2 \rfloor$ stages is $\frac{1}{4}$ of the maximum reward of the n-stage problem. These two results will be used to prove a 6-*competitive* ratio for the RDA in Theorem 6.

Lemma 5. *The expected reward that the RDA achieves by assigning the last $n - \lfloor n/2 \rfloor$ arriving tasks to the workers is at least $\frac{1}{8}$ times the maximum total reward.*

Proof. Assume that n is an even number. The proof for the case of n odd follows in a similar manner. The proof consists of two parts. The first part provides a lower bound on the reward achieved by the algorithm. The second part proves that the lower bound is at least half of the maximum offline reward. Combining these two parts yields the desired competitive ratio of 8.

The last $n/2$ tasks are assigned to the randomly selected $n/2$ workers in the selection set. Using the same notation as Theorem 5, let $P^* = \{p_1^*, p_2^*, \ldots, p_n^*\}$ denote the success rates of randomly ordered workers. For simplicity of notation, assume that $\{q_1, q_2, \ldots, q_{n/2}\}$ denote the first $n/2$ workers in the randomly ordered set, and $\{r_1, r_2, \ldots, r_{n/2}\}$ denote the last $n/2$ workers in the set. This means that $q_i = p_i^*$ for $1 \le i \le n/2$ and $r_j = p_{n/2+j}^*$ for $1 \le j \le n/2$. The RDA assigns the first $n/2$ arriving tasks to workers with success rates q_i and the last $n/2$ tasks to workers with success rates r_j.

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

Algorithm 3 (The Random Dividing Algorithm (RDA))

RDA $(n, P = \{p_i | i \in \{1, 2, \ldots, n\}\})$
Randomly order the workers $P^* = \{p_1^*, p_2^*, \ldots, p_n^*\}$
$A_{old} = P^*$
$s = \lfloor \log(n) \rfloor + 1$
for *j=1 to s* **do**
  $m = |A_{old}|$
  if $m > 1$ **then**
    $A_{new} = \{p_{(i)} \in A_{old} | i \in \{m - \lfloor m/2 \rfloor + 1, m - \lfloor m/2 \rfloor + 2, \ldots, m\}\}$
    $B(j) = A_{old} - A_{new}$
    $A_{old} = A_{new}$
  else
    $B(j) = A_{old}$
  end
end
$I_1 = (-\infty, +\infty)$
for *j=1 to s* **do**
  Assign the next r sequentially arriving tasks to workers with success rates $p_{(l)} \in B(s - j + 1)$, where $r = |B(s - j + 1)|$:
  if $x_i \in I_k$ **then**
    Assign the ith task to the worker with success rate $p_{(k)} \in B(s - j + 1)$ if it is not assigned before;
  end
  Update the intervals using the r tasks assigned:
  $I_k = (x_{(k)}, x_{(k-1)})$ for $k = 1, 2, \ldots, n + 1$, with $x_{(n+1)} = -\infty$ and $x_{(0)} = +\infty$
end

For the first part of the proof, notice that tasks are assigned to the workers according to algorithm B. Therefore, the expected reward of assigning the last $n/2$ tasks to the workers with success rates $\{r_1, r_2, \ldots, r_{n/2}\}$ is at least $\frac{1}{4}$ times the maximum offline reward (Lemma 2):

$$\sum_{j=1}^{n/2} x_{i_j} r_j \geq \frac{1}{4} \sum_{j=1}^{n/2} x_{(j)} r_{(j)}, \tag{16}$$

where x_{i_j} is the task assigned to worker with success rate r_j by the RDA, $x_{(j)}$ is the ith largest task value (among all n tasks), and $r_{(j)}$ is the jth largest success rate value in the set $\{r_1, r_2, \ldots, r_{n/2}\}$. Notice that in (16), the maximum offline reward refers to the maximum reward of the problem with workers $\{r_1, r_2, \ldots, r_{n/2}\}$ and all n tasks because we have assumed that by the random dividing, we only have access to this set of workers for the assignment.

Let $R_{max} = \sum_{i=1}^{n} x_{(i)} p_{(i)}$ denote the maximum offline reward of the n-stage problem. For the second part of the proof, we show that the lower bound provided by the right-hand side of (16) achieves a certain fraction of the maximum offline reward:

$$E\left[\sum_{j=1}^{n/2} x_{(j)} r_{(j)}\right] \geq \frac{1}{2} R_{max} \tag{17}$$

Because the workers are ordered randomly,

$$E\left[\sum_{j=1}^{n/2} x_{(j)} r_{(j)}\right] = \sum_{j=1}^{n/2} x_{(j)} E[r_{(j)}]$$

The probability that the jth largest success rate in the randomly selected set of workers (i.e., $r_{(j)}$) is the kth largest success rate among all n workers (i.e., $p_{(k)}$) is given by $\frac{\binom{k-1}{j-1}\binom{n-k}{n/2-j}}{\binom{n}{n/2}}$. Therefore, the expected value of $r_{(j)}$ is computed as $\sum_k \frac{\binom{k-1}{j-1}\binom{n-k}{n/2-j}}{\binom{n}{n/2}} p_{(k)}$. This yields

$$E\left[\sum_{j=1}^{n/2} x_{(j)} r_{(j)}\right] = \sum_{j=1}^{n/2} x_{(j)} \sum_{k=j}^{n/2+1} \frac{\binom{k-1}{j-1}\binom{n-k}{n/2-j}}{\binom{n}{n/2}} p_{(k)}. \tag{18}$$

Note that $x_{(j)}$ denotes the jth largest task value and the randomness is in the success rate of the worker assigned to the jth largest task. The constant factor multiplied by $p_{(k)}$ in (18) is given by

$$\sum_{j=1}^{k} \frac{\binom{k-1}{j-1}\binom{n-k}{n/2-j}}{\binom{n}{n/2}} = \frac{\binom{n-1}{n/2-1}}{\binom{n}{n/2}} = 1/2, \tag{19}$$

Therefore, (18) and (19) prove (17), and combining (17) and (16) yields the desired competitive ratio:

$$E\left[\sum_{j=1}^{n/2} x_{i_j} r_j\right] \geq \frac{1}{8} R_{max}. \quad \square \tag{20}$$

Lemma 6 focuses on the first $\lfloor n/2 \rfloor$ stages of the assignment process. Notice that whereas Lemma 5 provides a bound for the reward achieved by the RDA for the last $n - \lfloor n/2 \rfloor$ stages, Lemma 6 compares the maximum offline reward of the first $\lfloor n/2 \rfloor$ stages with the maximum offline reward of the n-stage problem. This means that Lemma 6 does not evaluate the performance of the RDA but instead finds the maximum (offline) expected reward of the first $\lfloor n/2 \rfloor$ stages when the workers are randomly ordered.

Lemma 6. *The maximum expected reward of the first $\lfloor n/2 \rfloor$ stages of the problem (i.e., with the first $\lfloor n/2 \rfloor$ tasks and the first $\lfloor n/2 \rfloor$ workers in the randomly ordered set) is $\frac{1}{4}$ times the maximum total reward of the n-stage problem.*

Proof. Assume that n is an even number. The proof for the case of n odd follows in a similar manner. We use the same notation as the previous lemma. Let $P^* = \{p_1^*, p_2^*, \ldots, p_n^*\}$ denote the success rates of randomly ordered workers, and $\{q_1, q_2, \ldots, q_{n/2}\}$ denote the first $n/2$ workers in the randomly ordered set. Let $\{x_1, x_2, \ldots, x_{n/2}\}$ denote the first $n/2$ tasks. Let $\hat{x}_{(j)}$ denote the jth largest task value among the first $n/2$ tasks, and $q_{(j)}$ is the jth largest success rate in the set $\{q_1, q_2, \ldots, q_{n/2}\}$. It must be proven that

$$E\left[\sum_{j=1}^{n/2} \hat{x}_{(j)} q_{(j)}\right] = \frac{1}{4} R_{max} = \frac{1}{4} \sum_{j=1}^{n} x_{(j)} p_{(j)}, \tag{21}$$

where $R_{max} = \sum_{j=1}^{n} x_{(j)} p_{(j)}$ is the maximum offline reward. Note that the left-hand side of (21) is the maximum offline reward of the first $n/2$ stages. The maximum offline reward assigns the jth largest task value among the first $n/2$ tasks to the worker with the jth largest success rate among the randomly selected workers. Because the workers and the task are (independently) ordered randomly,

$$E[\hat{x}_{(j)} q_{(j)}] = E[\hat{x}_{(j)}] E[q_{(j)}] = \sum_{k=j}^{n/2+1} \frac{\binom{k-1}{j-1}\binom{n-k}{n/2-j}}{\binom{n}{n/2}} x_{(k)} \sum_{k=j}^{n/2+1} \frac{\binom{k-1}{j-1}\binom{n-k}{n/2-j}}{\binom{n}{n/2}} p_{(k)}, \tag{22}$$

where $\frac{\binom{k-1}{j-1}\binom{n-k}{n/2-j}}{\binom{n}{n/2}}$ is the probability that the jth largest success rate (task value) in the randomly selected set of workers (among the first $n/2$ tasks) is the kth largest success rate (task value) among all n workers (tasks). Inserting (22) into (21),

$$E\left[\sum_{j=1}^{n/2} \hat{x}_{(j)} q_{(j)}\right] = \sum_{j=1}^{n/2} E[\hat{x}_{(j)}] E[q_{(j)}] = \sum_{j=1}^{n/2} \left[\sum_{k=j}^{n/2+1} \frac{\binom{k-1}{j-1}\binom{n-k}{n/2-j}}{\binom{n}{n/2}} x_{(k)} \sum_{k=j}^{n/2+1} \frac{\binom{k-1}{j-1}\binom{n-k}{n/2-j}}{\binom{n}{n/2}} p_{(k)}\right]. \tag{23}$$

Using (19), the constant factor multiplied by $p_{(k)}$ ($x_{(k)}$) in (23) is $\frac{1}{2}$. Therefore, the constant factor multiplied by $x_{(k)} p_{(k)}$ is $\frac{1}{4}$, which completes the proof. $\square$

Note that the result of Lemma 6 is intuitive. The expected reward of the first $\lfloor n/2 \rfloor$ tasks is $\frac{1}{2}$ of the expected reward of n tasks because they arrive in a random order. When the workers are also ordered randomly, the maximum expected reward of the first $\lfloor n/2 \rfloor$ stages become $\frac{1}{4}$ of the maximum reward of the n-stage problem.

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.

Table 1. The Performance of the RIRA, DDA, and RDA for the GSSAP with Uniform Task Values

No. of tasks	$\min(R_{RIRA}/R_{opt})$	$\min(R_{DDA}/R_{opt})$	$\min(R_{RDA}/R_{opt})$
10	55.4%	63.1%	57.6%
50	72.7%	77.7%	67.8%
100	72.9%	80.5%	71.2%

Theorem 6. *The RDA is 6-competitive.*

Proof. Let $\{p_1^*, p_2^*, \ldots, p_{l^k}^*\}$ denote the success rates of the workers assigned at the kth round with $l^{(k)} = \lfloor \frac{1}{2} \ldots \lfloor \frac{1}{2} \lfloor \frac{n}{2} \rfloor \rfloor \ldots \rfloor$ (with k multipliers of $\frac{1}{2}$). Let x_i^* and x_i^{opt} denote the values of the tasks assigned to the worker with success rate p_i^* by the RDA and the optimal policy, respectively. Then, by Theorem 2,

$$\sum_{i=1}^{l^k} x_i^* p_i^* \geq \frac{1}{4} \sum_{i=1}^{l^k} x_i^{opt} p_i. \tag{24}$$

Using Lemma 5, each assignment round achieves $\frac{1}{8}$ of the maximum reward of that round, and using Lemma 6, the maximum reward of each round is $\frac{1}{4}$ of the maximum reward of the previous round. Therefore,

$$E[R_{RDA}] = E\left[\sum_{l^k} \sum_{i=1}^{l^k} x_i^* p_i^*\right] \geq \frac{1}{4} E\left[\sum_{l^k} \sum_{i=1}^{l^k} x_i^{opt} p_i\right] \geq \frac{1}{4}\left(\frac{1}{2} + \frac{1}{8} + \ldots\right) R_{max}, \tag{25}$$

where $E[R_{RDA}]$ denotes the expected reward achieved by the RDA, and $R_{max} = \sum_{i=1}^{n} x_{(i)} p_{(i)}$ is the maximum offline reward. Therefore, as $n \to +\infty$, the RDA is 6-*competitive*. □

3.4. Optimal Competitive Ratios

The proposed algorithms for GSSAP are not optimal. The linear programming technique can be used to provide an upper bound on the performance of the optimal policies for online matching problem. The linear programming technique models the online matching problem as a linear program, with the objective providing an upper bound on the expected reward and the constraints representing the problem constraints. For example, the constraints of GSSAP guarantee that each task is assigned to one worker and each worker is assigned to one task. This technique can be used to prove the following result (Buchbinder et al. 2013).

Theorem 7. *No algorithm for GSSAP can achieve a competitive ratio better than e. Moreover, no algorithm for the incentive compatible problem can achieve a competitive ratio better than 4.*

4. Numerical Experiments

This section reports the results of several numerical experiments. The experiments use the RIRA (RIRA), deterministic dividing algorithm (DDA), and the random dividing algorithm (RDA) to assign sequentially arriving tasks to workers with fixed success rates. Table 1 shows the results for task values generated uniformly with values between $(1, 10)$, whereas Table 2 reports the results for exponentially distributed task values with mean 10. The workers success rates for the GSSAP with n tasks are $\{1, 2, \ldots, n\}$ (i.e., the success rate of the jth worker is j). The performance of the three methods is compared by reporting the minimum ratio between the online and offline rewards in 1,000 iterations (i.e., the minimum ratio between the reward achieved by each of the algorithms and the maximum offline reward over 1,000 different problem instances).

In all experiments DDA performs the best. DDA assigns half of the workers with the largest success rates (i.e., the second, fourth, sixth, …, largest success rates among all n workers) to the first $\lfloor n/2 \rfloor$ arriving tasks and the other half to the last $n - \lfloor n/2 \rfloor$ arriving tasks. Because the tasks have a random arrival order, the ith largest task value is

Table 2. The Performance of the RIRA, DDA, and RDA for the GSSAP with Exponential Task Values

No. of tasks	$\min(R_{RIRA}/R_{opt})$	$\min(R_{DDA}/R_{opt})$	$\min(R_{RDA}/R_{opt})$
10	32.3%	39.5%	28.9%
50	50.6%	59.6%	55.0%
100	60.0%	70.0%	55.9%

equally likely to arrive in the first half or the second half of the assignment process. Therefore, DDA increases the expected number of assignments in which the worker with the *i*th largest success rate is assigned to the task with the *i*th largest value and hence, increases the total expected reward.

5. Summary and Conclusion

This paper describes the sequential stochastic assignment problem (SSAP) as a generalization of the secretary problem, in which each of the selected elements is assigned to a distinct position. The weighted secretary problem is used to relax the assumption that task values are independently drawn from a known distribution in SSAP and propose assignment policies for the generalized SSAP (GSSAP). The proposed assignment policies provide intuitive models to deal with the training phase in an online matching problem. Exploring new extensions of GSSAP using the secretary problem and studying the relation between the GSSAP and online matching problems are possible future research directions.

Acknowledgment

The authors thank the Associate Editor and the reviewers for their insightful comments, resulting in a significantly improved manuscript.

References

Albright SC (1974) Optimal sequential assignment with random arrival time. *Management Sci.* 21(1):60–67.
Albright SC, Derman C (1972) Asymptotic optimal policies for stochastic sequential assignment problem. *Management Sci.* 19(1):46–51.
Babaioff M, Dinitz M, Gupta A, Immorlica N, Talwar K (2009) Secretary problems: Weights and discounts. *Proc. 20th Annual ACM-SIAM Sympos. Discrete Algorithms, SODA '09* (Society for Industrial and Applied Mathematics, Philadelphia), 1245–1254.
Buchbinder N, Jain K, Singh M (2013) Secretary problems via linear programming. *Math. Oper. Res.* 39(1):190–206.
Chow YS, Moriguti S, Robbins H, Samuels SM (1964) Optimal selection based on relative rank. *Israel J. Math.* 2(2):81–90.
Derman C, Lieberman GJ, Ross SM (1972) A sequential stochastic assignment problem. *Management Sci.* 18(7):349–355.
Enns EG (1970) The optimum strategy for choosing the maximum of n independent random variables. *Unternehmensforschung* 14(1):89–96.
Freeman PR (1983) The secretary problem and its extensions: A review. *Internat. Statist. Rev.* 51(2):189–206.
Gershkov A, Moldovanum B (2010) Efficient sequential assignment with incomplete information. *Games Econom. Behav.* 68(1):144–154.
Gianini J, Samuels SM (1976) The infinite secretary problem. *Ann. Probab.* 4(3):418–432.
Glasser KS, Holzsager R (1983) The d choice secretary problem. *Comm. Statist.* 2(3):177–199.
Goel G, Nikzad A, Singla A (2014) Allocating tasks to workers with matching constraints: Truthful mechanisms for crowdsourcing markets. *Proc. 23rd Internat. Conf. World Wide Web* (ACM, New York), 279–280.
Kennedy DP (1986) Optimal sequential assignment. *Math. Oper. Res.* 11(4):619–626.
Khatibi A, Jacobson SH (2014) Doubly stochastic sequential assignment problem. *Naval Res. Logist.* 63(2):124–137.
Khatibi A, Baharian G, Kone ER, Jacobson SH (2014) The sequential stochastic assignment problem with random success rates. *IIE Trans.* 46(11): 1169–1180.
Kleinberg R (2005) A multiple-choice secretary algorithm with applications to online auctions. *Proc. 16th Annual ACM-SIAM Sympos. Discrete Algorithms, SODA '05* (Society for Industrial and Applied Mathematics, Philadelphia), 630–631.
Lindley DV (1961) Dynamic programming and decision theory. *Appl. Statist.* 10(1):39.
Nikolaev AG, Jacobson SH (2010) Stochastic sequential decision-making with a random number of jobs. *Oper. Res.* 58(4P1):1023–1027.
Oveis Gharan S, Vondrák J (2011) On variants of the matroid secretary problem. Demetrescu C, Halldórsson MM, eds. *Algorithms – ESA 2011.* Lecture Notes in Computer Science, vol. 6942 (Springer, Berlin, Heidelberg), 335–346.
Presman EL, Sonin IM (1972) The best choice problem for a random number of objects. *Theory Probab. Appl.* 17(4):657–668.
Rasmussen WT, Pliska SR (1975) Choosing the maximum from a sequence with a discount function. *Appl. Math. Optim.* 2(3):279–289.
Sakaguchi M (1978) Dowry problems and ola policies. *Rep. Statist. Appl. Res. JUSE* 25:124–128.
Smith MH (1975) A secretary problem with uncertain employment. *J. Appl. Probab.* 12(03):620–624.
Su X, Zenios SA (2005) Patient choice in kidney allocation: A sequential stochastic assignment model. *Oper. Res.* 53(3):443–455.

Downloaded from informs.org by [216.73.217.61] on 14 July 2026, at 07:50 . For personal use only, all rights reserved.