Stochastic Scheduling: Strategies for Abandonment Management

RICE UNIVERSITY
Georgia Institute of Technology

Yihua Xu[1]; Rohan Ghuge[2]; Sebastian Perez-Salazar[1]

[1]Computational Applied Mathematics & Operations Research, Rice University; [2]Industrial and Systems Engineering, Georgia Tech

MIXED INTEGER PROGRAMMING WORKSHOP

Motivation

Problem: Revenue-maximizing multi-stage stochastic scheduling problems comprising two sources of ***uncertainty***

- Jobs have a stochastic service time
- Impatient customers may leave at a random time

The Model

- Single server (idle/busy), discrete time
- n jobs with values $v_1, \dots v_n > 0$
- Unknown stochastic last available time D_i
- Unknown stochastic service time S_i

Dynamic

- At each time t, if server is idle, then we can run an *available* job and obtain a value of v_i
- Server remains busy for S_i units of time

Time (t)	1	$1+S_i$	$1+S_i+S_{i'}$	$\dots$
Available Jobs	$[n]$	R_{1+S_i}	$R_{1+S_i+S_{i'}}$	
Job Run	i	i'	i''	
Reward up to time t	v_i	$v_i+v_{i'}$	$v_i+v_{i'}+v_{i''}$	

Goal

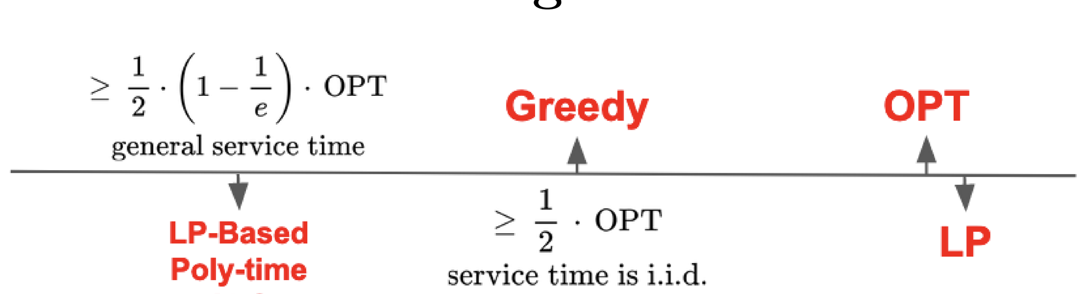
Aim to find policy/algorithm ALG that maximizes:

$$\mathbf{E}[V_{ALG}] \doteq \mathbf{E}\left[\sum_{i \text{ run by } ALG} v_i\right]$$

- Can be solved via Dynamic Programming
  - An **NP-Hard** problem
- Curse of Dimensionality
- Find **Approximate** Solution!
  - Approx. ratio α for a maximization problem is defined as:

$$\alpha = \min_I \left(\frac{V_{\texttt{ALG}}(I)}{\texttt{OPT}(I)}\right)$$

Looking Ahead



IID - Greedy

Theorem 1

When service times are IID, greedy policy that runs the highest-valued available job whenever the server is free guarantees at least 1/2 of the optimum.

- *Coupling* - Server free at same time step.

Time (t)	1	2	3	4	5	6	7	8	9	10	$\dots$
Greedy Policy	✓			✓	✓					✓	
Optimal policy	✓			✓	✓					✓	

- *Charging* - When Greedy serves a job:

$$2 \cdot \text{Greedy}_t \geq \texttt{OPT}_t + \texttt{OPT}_{t'>t}$$

Time (t)	t	t'
Greedy Policy	run $j \to 2\cdot v_j$	≥ 0
Optimal policy	run $i \neq j \to v_i$	run $j \to v_j$

Example - Greedy Fails in General

Consider the following n jobs. Job 1:

$$v_1 = 1+\varepsilon, D_1 = n+1, S_1 = n$$

For each job $i \in \{2, \dots, n\}$, we have:

$$v_i = 1, D_i = n, S_i = 1$$

Then: $\mathbb{E}[V_{\text{Greedy}}] = 1+\epsilon$ v.s. $\mathbb{E}[\texttt{OPT}] = n-1$

LP Bound and Algorithm

We can formulate the problem by using an LP where the variable $x_{i,t}$ denotes the Pr[OPT runs i at t]. We maximize the expected value, i.e.

$$v_{LP} = \max_{x \geq 0} \quad \sum_{t=1}^{T}\sum_{i=1}^{n} v_i x_{i,t}$$

The problem is subject to following two constraints:

- Every job is run at most one time

$$\sum_{t=1}^{T} \frac{x_{i,t}}{\Pr(D_i \geq t)} \leq 1 \qquad \forall i \in [n]$$

- Each time is busy by at most one job

$$\sum_{i=1}^{n} x_{i,t} + \sum_{i=1}^{n}\sum_{\tau=1}^{t-1} x_{i,\tau} \Pr(S_i > t-\tau) \leq 1 \qquad \forall t \in [T]$$

Theorem 2

The value of the LP, $V_{LP} \geq V(\texttt{OPT})$. Moreover, there is an efficiently computable algorithm ALG that guarantees $\mathbb{E}[V_{\texttt{ALG}}] \geq 1/2 \cdot (1-1/e-\epsilon) \cdot V(\texttt{OPT})$, under mild assumptions on service time.

LP-Based Algorithm

Denote $f_{i,t}$: Probability that all three events hold:
① job i not considered before t;
② job i not departed by t;
③ server idle at t

We can describe algorithm ALG as the following:

- Given any instance I, find x^* solution to LP
- For each $t = 1, 2, \dots$ that server is idle
  1. If job i not considered before t and available pick it with probability $x^*_{i,t}/(2 \cdot f_{i,t})$
  2. Run highest-valued available job picked above

Proof Intuitions

We show that for each time horizon t,

$$\frac{\mathbf{E}[V_{ALG,t}]}{\sum_{i=1}^{n} v_i x_{i,t}} \geq \frac{1}{2}\left(1-\frac{1}{e}\right)$$

- **Factor** 1/2: For all time t -

$$\Pr[\text{Server is free at time } t] \geq \frac{1}{2}$$

- **Factor** $1-1/e$: Suppose each job is considered with probability p_i, we can find the following relationship between the greedy choice and the LP average for all time t -

$$\mathbf{E}\left[\max_{i \in [n]}\{X_i\}\right] \geq \left(1-\frac{1}{e}\right)\sum_{i=1}^{n} v_i p_i$$

Further Analysis

Prop 1 - Tightness on Approx. Ratio

There exists an instance I such that:

$$V_{\texttt{ALG}}(I) \leq (1-1/\sqrt{e}+\epsilon)V_{LP}(I)$$

Prop 2 - Upper Bound on Approx. Ratio

There exists an instance I such that:

$$\texttt{OPT}(I) \leq (1-1/e+\epsilon)V_{LP}(I)$$

Flexibility

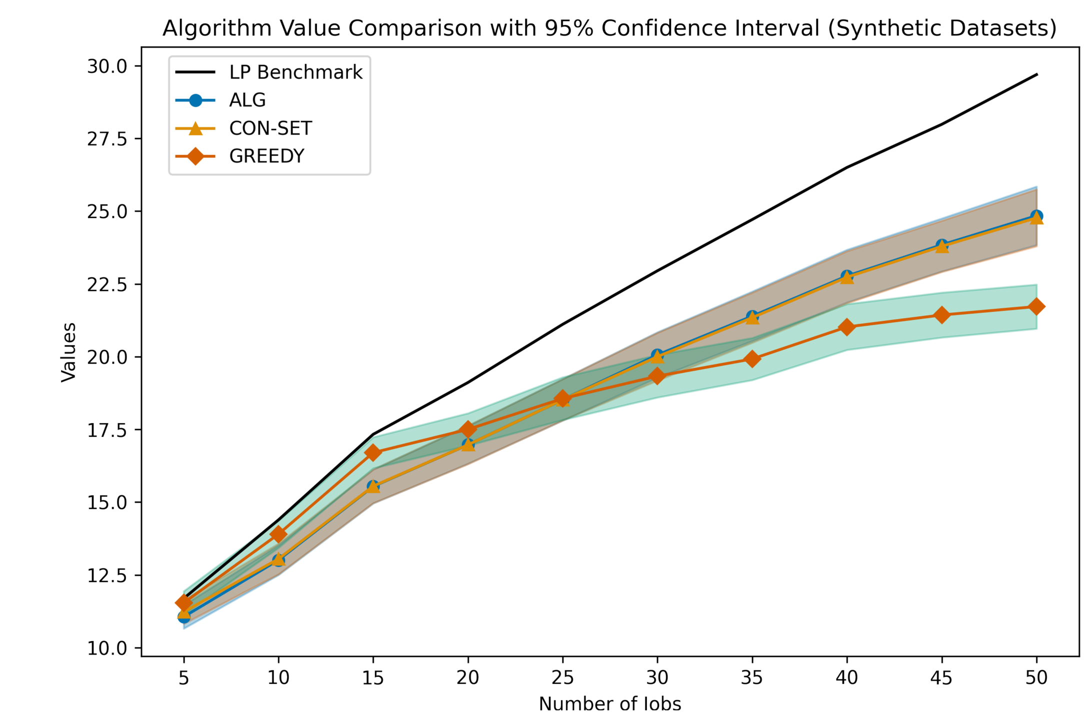
Our algorithm is flexible to include deadlines, knapsack, and cardinality constraints with modified approx. ratio.

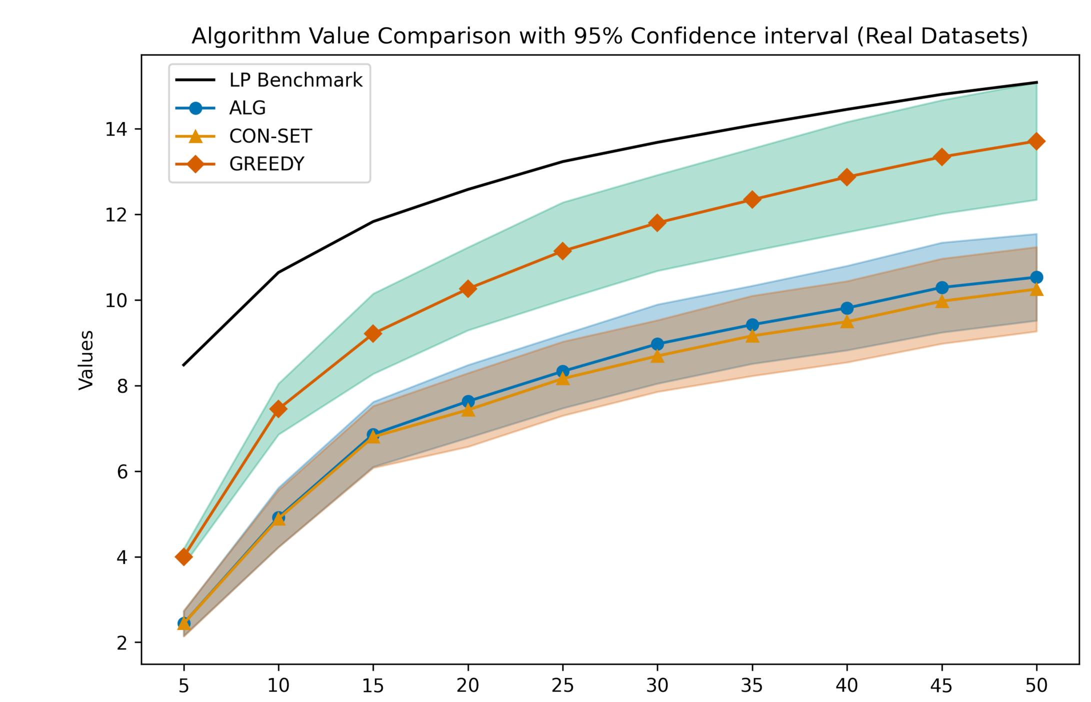
Problems	Approximation Ratio
Deadline	$1/2 \cdot (1-1/e)$
Knapsack	$1/2 \cdot (1-1/e) \cdot \left(1-e^{-B^2/(2nw_{max}^2)}\right)$
Cardinality	$1/2 \cdot (1-1/e) \cdot \left(1-e^{-k/6}\right)$

Table 1:We use w_{max} to denote $\max_{i \in [n]} w_i$. B and k each represent the size of knapsack and cardinality respectively.

Numerical Experiments

We test algorithms using both synthetic and real dataset from an anonymous Israel bank call center[1].





- ALG attains a high competitive ratio with consistent performances.

Instance Type	ALG-S	ALG-E	ConSet	Greedy
Syn-5	8.07	0.65	1.04	0.57
Syn-25	76.82	4.56	5.92	4.08
Syn-50	614.82	18.27	26.10	14.06

Table 2:Runtime in select synthetic datasets. Column ALG-S represents the simulation time needed to retrieve $f_{j,t}$ values, and the column ALG-E) is the execution part to obtain values.

- ALG takes considerably longer time, primarily due to the simulations needed for retrieving $f_{i,t}$ values. Another method (named ConSet):
  - Leverage the structure of forming a consideration set
  - No simulations needed - diminish runtime
  - Comparable performance

Future Work

- Hardness (#P, APX, PSPACE, etc)
- Multiple Server & Multiple Resources
- Arrivals

[1]Data can be accessed from https://seelab.net.technion.ac.il/data/