Stochastic Dynamic Optimization for Dynamic Scheduling

Yasin Göçgün
Associate Professor of Industrial Engineering
Faculty of Engineering and Natural Sciences
İstinye University
İstanbul, Turkey
yasin.gocgun@istinye.edu.tr

Abstract

We study two classes of dynamic scheduling problems termed as "allocation" and "advanced" scheduling. In allocation scheduling, arriving jobs either wait in queue, are rejected, or served immediately, while in advanced scheduling, they are scheduled to time slots such as days in a booking horizon. We develop approximate dynamic programming (ADP) based on direct search that approximately solves the underlying Markov decision process models. We compare the performance of the proposed technique against the myopic policy under various scenarios. Numerical results demonstrate that the direct-search based ADP yields significant improvements over the myopic policy in all of the problem sets.

Keywords
dynamic stochastic scheduling; Markov decision processes; direct search

1. Introduction

Scheduling problems that have the dimension of dynamicity arise in many fields such as manufacturing, communication, and transportation. Differing from static scheduling problems, those problems contain jobs arriving randomly at the system. In manufacturing settings, jobs correspond to orders or tasks, whereas in healthcare settings, patient requests can be considered as jobs (Göçgün & Puterman, 2014). In communication networks, requests for data or video transmission are viewed as jobs, and the resource corresponds to bandwidth (Altman, 2002).

In a class of dynamic scheduling problems, termed as allocation scheduling, arriving jobs that are reviewed periodically may be served immediately or are rejected/diverted, with the goal of optimizing some performance metric. Another class of dynamic scheduling problems, termed as advanced scheduling, has the property that jobs arriving at the underlying system are scheduled to epochs within a booking horizon, taking into account resource constraints. Costs such as delay costs and penalty cost of rejection are incurred. One extension of these problems contains a waiting list, through which decisions can be postponed, putting some of the arriving requests on a waiting list. This gives the planner additional flexibility, improving service quality. The planner incurs a holding cost for each type-i job that stays on the waiting list for one period. In another extension, jobs that are scheduled in a booking horizon require multiple resources and are likely to be cancelled and hence must be rescheduled, as observed in chemotherapy scheduling where patient requests are viewed as jobs (Göçgün, 2018).

The abovementioned classes of scheduling problems are termed as dynamic stochastic scheduling (DSS) problems (Göçgün & Ghate, 2012). This is due to the fact that these problem have dynamic features because of dynamic arrivals of jobs; stochasticity is observed owing to random arrivals of jobs.

To this end, we consider two classes of DSS problems: allocation scheduling problems and advanced scheduling problems, the latter involving the following features: jobs requiring multiple resources, and waiting list for arriving jobs. In line with the literature, we provide an approximate solution to the underlying mathematical models of these problems. Specifically, we develop an approximate dynamic programming technique centered on direct-search to approximately solve the Markov decision process (MDP) models of these problems.

2. Literature Review
Literature on dynamic scheduling is quite rich. We provide a brief summary of the papers from allocation scheduling and advance scheduling next.

2.1 Literature Review on Allocation Scheduling
Göçgün & Ghate (2012) studied dynamic stochastic allocation scheduling problems taking into account multiple resources. They developed a Markov Decision Process (MDP) model of these problems and solved them approximately via a Lagrangian-based Approximate Dynamic Programming (ADP) technique.

Huh et al. (2013) considered a multi-resource allocation scheduling problem considering elective and emergency patients. The authors discussed convexity of their formulation and developed a heuristic policy called "limit" policy for solving the underlying problem.

Barz & and Rajaram (2015) studied an allocation scheduling problem in hospital settings taking into account emergency patients as well as elective patients who must be admitted to a hospital or are rejected. Formulating the problem as an MDP, the authors employed ADP to obtain bounds and developed heuristics.

It is worth noting that a detailed literature on dynamic stochastic allocation scheduling problems can be found in Göçgün & Ghate (2012), which includes the mathematical model we consider in our work.

2.2 Literature Review on Advanced Scheduling
Patrick et al. (2008) proposed an MDP formulation of problems where patients of distinct types are scheduled to time slots. Owing to computational intractability, they resorted to ADP techniques, developing a linear-programming based ADP technique for approximately solving the respective MDP.

Saure et al. (2012) extended the work in Patrick et al. (2008) by a similar but slightly complicated problem that arises in radiation therapy units. They formulated the problem using MDPs and proposed a linear programming-centered ADP for solving the respective MDP.

Göçgün & Ghate (2012) studied a class of scheduling problems, considering multiple resources. They formulated the underlying problem using MDPs and developed a Lagrangian relaxation technique to approximately solve the underlying MDP model.

Ceschia and Schaerf (2016) focused on a patient admission scheduling problem, taking into account constraints on the utilization of operating rooms. Their model contains features such as flexible planning horizon and new components of the objective function. The authors developed a local search for solving this problem.

Wang et al. (2015) worked on web applications used to schedule advance service, considering non-stationary arrivals. The authors formulated these problems as online weighted bipartite matching problems and developed algorithms with performance guarantees.

Göçgün & Puterman (2014) studied a chemotherapy scheduling problem that considers patients of different types who have target dates as well as tolerance limits. They provided an MDP formulation of the problem and developed an ADP centered on linear-programming for obtaining an approximate solution. Göçgün (2018) extended the model proposed by Göçgün & Puterman (2014) by including cancellation of jobs. The author provided an MDP formulation of the problem and approximately solved the problem using ADP. In a related work, Abdırahman (2019) studied a class of dynamic scheduling problems that have more generic features, considering appoointments without target dates and time windows.

Finally, Göçgün (2021) compares a variety of ADP techniques such as Lagrangian-based ADP and ADP centered on direct search under diverse settings for dynamic scheduling problems. Their results demonstrated that ADP utilizing direct searcch performs generally better than other APP methods.

In this work, we extend the problems studied in Göçgün & Ghate (2012) and Patrick et al. (2008) by including extensions such as waiting list for jobs.

3. Dynamic stochastic allocation scheduling

We address a type of allocation scheduling problems introduced in Göçgün & Ghate (2010). We provide a brief description of these problems below.

• We consider heterogeneous job types.
• Arrivals of jobs to the system are random.
• The job arrival process for each type is assumed to be independent.
• We consider multiple resource constraints.
• Each job that arrives must be chosen for service or be rejected.
• Service time for each job is assumed to be one-time period.
• A reward is received when a job is completed.
• A penalty cost is incurred for a job that is rejected.
• There is a seperate type-dependent queue capacity.
• The objective is "to decide which of the arriving jobs to select for service in each period so as to maximize the total discounted expected profit over an infinite horizon". (Göçgün & Ghate, 2010)

3.1 Mathematical model

The aforementioned problem is formulated using Markov Decision Processes (MDPs) (see Göçgün & Ghate, 2010 for an equivalent formulation).

State Space: $x = (x_1, x_2, \dots, x_I)$, where x_i is the number of jobs from type i in queue at the start of a time-period for $i = 1,2, \dots, I$.

The Action Set: $y = (y_1, y_2, \dots, y_I)$, where y_i is the number of type- i jobs that will be chosen for service in a time-period for $i = 1,2, \dots, I$,. We define $Y(x)$ as the Cartesian product $Y_1(x_1) \times Y_2(x_2) \times \dots \times Y_I(x_I)$, where $Y_i(x_i) = \{0,1, \dots, x_i\}$ (Göçgün & Ghate, 2010). The set $\bar{Y}(x) \subseteq Y(x)$ of all actions that are feasible in state x is defined by the following resource constraints:

$$\bar{Y}(x) = \{(y_1, y_2, \dots, y_I) \in Y(x): \sum_{i=1}^{I} a_{ij} y_i \leq b_j,\ j = 1,2, \dots, J\}, \tag{1}$$

where a_{ij} is the amount of resource j required for performing a job from type-i, b_j is total amount of resource j that is available in each time-period.

The discounted expected profit for the underlying period is expressed as (Göçgün & Ghate, 2010)

$$f(x,y) = \lambda \sum_{i=1}^{I} R_i y_i - \sum_{i=1}^{I} H_i(x_i - y_i) - \lambda \sum_{i=1}^{I} \sum_{n_i=0}^{K_i} p_i(n_i) G_i(\max\{(x_i - y_i) + n_i - W_i, 0\}), \tag{2}$$

where λ is discount factor, R_i is a reward received upon completing a job from type-i, H_i is holding cost incurred for a job from type-i per period, K_i is the maximum number of jobs from type-i that may arrive during a time period, $p_i(n_i)$ is the probability that n_i jobs from type-i arrive during a time-period, G_i is a penalty cost incurred for rejecting a job from type-i per time-period, and W_i is queue capacity for a type-i job. It is easy to see that $f(x,u)$ consists of terms corresponding to reward, holding cost, and penalty cost.

$f(x,y)$ can be expressed as $\sum_{i=1}^{I} f_i(x_i, y_i)$ (Göçgün & Ghate, 2010), where

$$f_i(x_i, y_i) = \lambda R_i(y_i) - H_i(x_i - y_i) - \lambda \sum_{n_i=0}^{K_i} p_i(n_i) G_i(\max\{(x_i - y_i) + n_i - W_i, 0\}). \tag{3}$$

The resulting Bellman's equations are given by

$$V(x) = \max_{y \in \bar{Y}(x)} \{\sum_{i=1}^{I} f_i(x_i, y_i) + \lambda \sum_{n_1=0}^{K_1} \dots \sum_{n_I=0}^{K_I} (\prod_{i=1}^{I} p_i(n_i)) V(x'_1, x'_2, \dots, x'_I)\}, \tag{4}$$

where

$x'_i = \min\{(x_i - y_i) + n_i, W_i\},$ for $i = 1,2, \dots, I$.(5)

Because of computational intractability, Bellman's equations cannot be solved exactly for large-sized problems. This forces us to utilize an approximate dynamic programming (ADP) method for solving the underlying mathematical model approximately.

3.2 Direct search-based ADP

A variety of intractable MDPs in diverse fields have been solved through ADP techniques (Powell, 2007). These techniques are categorized as ADPs centered on linear programming (LP) and ADPs centered on simulation (see Adelman, 2003 and Adelman, 2004 for detailed information about these types of ADPs).

The implementation of ADP requires that basis functions possessing certain crucial features of the states of the underlying system are utilized for approximating the value function. One example of making use of basis functions is linear approximation, which is expressed as follows:

$$V(s) \approx \sum_{k=1}^{K} r_k \Phi_k(s),$$

where r_k, k=1,…,K represent parameters that are tuned and $\Phi_k(s)$, k=1,…,K symbolize basis functions (Göçgün, 2021). Those parameters are tuned in an iterative way to retrieve an ADP policy after approximating the value function is completed. In this regard, ADP approaches seek the optimal parameter vector that eanbles to minimize a certain performance metric. Regression-based techniques are typically used to solve the underlying optimization problem (Powell, 2007). We however use a direct search-based technique that tunes the ADP parameters to reach the best policy. The ADP policy is then acquired using the approximate value functions (Göçgün, 2021).

3.2.1 Retrieving the ADP Policy

As stated earlier, the parameter tuning phase leads to the approximate value of the underlying state. Then, we obtain the ADP policy via the computation of an action vector for any state the system visits (Göçgün, 2021). The following expression gives us the decision retrieval problem for a given state s of our MDP model:

$$\max_{y} \{\sum_{i=1}^{I} [f_i(x_i, y_i) + \lambda \sum_{n_i=0}^{K_i} p_i(n_i)\tilde{v}(x')]\} \tag{6}$$
$$y_i \in Y_i(x_i) = \{0,1,\dots,x_i\},\ i = 1,2,\dots,I;$$
$$\sum_{i=1}^{I} a_{ij} y_i \le b_j,\ j = 1,2,\dots,J,$$

where $\tilde{V}(x')$ is the approximate value of state x'.

3.2.2 Basis Functions

We choose the following basis functions for our ADP implementation:

$$\Phi_j(x) = \sum_{i=1}^{I} a_{ij}(y_i), j \in J$$

According to the above basis functions, the value of a state increases with an increase in amount of resource consumed.

3.2.3 Direct Search

In our ADP implementation, parameters are tuned using direct search, which aims to find good policies. Specifically, an optimization problem where feasible r's represent the variables and the objective is to minimize the the expected cost of the underlying policy is solved by direct search. The underlying optimization problem is given as

$$\min_{r \in R^N} \sum_{t=0}^{\infty} c(s_t, \pi_r(s_t)), \tag{7}$$

where s_t is the state at stage t of the system, π_r is "the policy obtained by the parameter vector r, $\pi_r(s_t)$ is the action dictated by the policy π_r in the state at stage t, and $c(s_t, \pi_r(s_t))$ is immediate cost incurred at step t as a result of choosing $\pi_r(s_t)$." (Göçgün, 2021).

While implementing the ADP approach centered direct search, we let $r_j, j \in J$ range from 2 to 50 in increments of 2.

3.3 Numerical Results

Data generation process was in line with the related literature (Göçgün & Ghate, 2010). Number of job types was set to 5 and 10. Number of distinct resources was set to 2. a_{ij} was set to $i + \lceil 2 \times i \times U(0,1) \rceil$, and we set b_j to $\lceil t.r \times \sum_{i=1}^{I} a_{ij} \rceil$, where $t.r$ is called tightness ratio (Göçgün & Ghate, 2010). $t.r$ was set to $\{0.5, 0.75, 1, 1.25\}$. We set reward for type-i job to $50i + DU(1,50)$ (DU stands for discrete uniform). Holding cost and penalty cost for type-i job were set to $DU(1,50)$ and $DU(1,100)$, respectively. Queue capacity for each job type was set to 3. We set λ to 0.8 and 0.99.

Simulation run length and number of replications were set to 50 and 20, respectively. For each combination of λ, I, and $t.r$, we solved 10 problem instances.

We compare the ADP centered on direct search against the myopic policy in terms of total profit. The myopic policy is acquired by solving the problem $\max_{y \in Y} f(x,y)$ for any state $x \in X$. We utilized AMPL, CPLEX 12 for coding the algorithms and solving all integer programs during the implementation of the algorithms.

Results are shown in Tables 1 and 2. Each table contains the corresponding values of $t.r$ and λ and the profits obtained by the direct search-based ADP and the myopic policy as averages of 200 values for each problem set(owing to having 20 replications and 10 problem instances for each problem set). The last column provides percentage improvement over the myopic policy. Average percentage improvements over the myopic policy for $I = 5$ and $I = 10$ are 19% and 16%, respectively.

Table 1. Results for allocation scheduling with $I = 5$.

$(t.r, \lambda)$	DS-ADP	Myopic	Per. impr.
(0.5,0.9)	-2198	-2448	10.2
(0.5,0.99)	-15707	-17426	9.8
(0.75,0.9)	-239	-590	59.3
(0.75,0.99)	-2802	-5072	44.7
(1,0.9)	2063	1874	10.1
(1,0.99)	13035	11813	10.3
(1.25,0.9)	3999	3889	2.8
(1.25,0.99)	27341	26278	4

Table 2. Results for allocation scheduling with $I = 10$.

$(t.r, \lambda)$	DS-ADP	Myopic	Per. impr.
(0.5,0.9)	1903	1308	37.8
(0.5,0.99)	12667	8342	51.8
(0.75,0.9)	9462	8482	11.5
(0.75,0.99)	64941	57128	13.7
(1,0.9)	16826	16156	4.1
(1,0.99)	116551	112149	3.9
(1.25,0.9)	24441	24006	1.8
(1.25,0.99)	171661	167881	2.2

3.4 Discussion

The abovementioned results lead us to the following inferences. When tightness ratio is reasonably small (being less than 1), the percentage improvement is significantly high in both cases. The higher tightness ratio is, the smaller percentage improvement will be. This implies that the impact of the direct search-centered ADP is more prominent in the case of scarce resource. Further, the percentage improvement generally decreases with an increase in discount factor when $I = 5$, while it generally increases in the same situation when I is higher. It is also worth noting that the impact of the direct search-based ADP is slightly higher when number of types is small (corresponding to the I=5 case).

4. Dynamic stochastic advanced scheduling

The features of the class of stochastic advanced scheduling problems we study are provided below(see and Göçgün & Ghate, 2012 for a similar description).

• There are heterogeneous job types.
• We consider random arrivals of jobs to the system.
• Each job that arrives must be scheduled to a day within a booking horizon. Jobs can be rejected (or outsourced or served through overtime).
• Jobs of each type are associated with a deadline. Delay cost is incurred if an arriving job is scheduled to a day after its deadline.
• A penalty cost is incurred for jobs that are rejected.
• The objective is to perform scheduling and rejection for jobs that arrive so as to "minimize the total discounted expected cost over an infinite horizon" (Göçgün & Puterman, 2014).

4.1 Dynamic stochastic advanced scheduling with multiple resources and waiting list

In addition to the abovementioned features, this class of problems have the following properties:

• We consider multiple resource constraints.
• Jobs that are neither scheduled nor rejected are held in a waiting list.

4.1.1 Mathematical model

We formulate the abovementioned problem using Markov Decision Process (MDP), the components of which are given below (see Göçgün & Ghate, 2012 for an equivalent MDP model in terms of state space and action set).

State Space: $s = (x, y) = (x_{in}, y_i)$, $i = 1, \dots, I$ and $n = 1, \dots, N$, where where I is the number of job types, N is the length of the booking horizon, x_{in} for $i = 1, \dots, I$, $n = 1, \dots, N$ is number of type-i jobs that are already scheduled to day n, y_i for $i = 1, \dots, I$ is number of type-i jobs awaiting for scheduling. Note that the state set S must be defined as follows:

$$S = \{(x, y) | \textstyle\sum_{i=1}^{I} a_{ij} x_{in} \leq b_j, j = 1, \dots, J, n = 1, \dots, N\},$$

where a_{ij} is an amount of resource $j \in J$ required for performing each job of type-i; b_j is total amount of resource $j \in J$ that is available each day.

The Action Set: $(u, z) = (u_{in}, z_i)$, $i = 1, \dots, I$ and $n = 1, \dots, N$, where u_{in} for $i = 1, \dots, I$ and $n = 1, \dots, N$ is number of type-i jobs scheduled to day n, and z_i for $i = 1, \dots, I$ is number of type-i jobs that are rejected. The below constraints must be satisfied by any action:

$$\textstyle\sum_{i=1}^{I} a_{ij}(x_{in} + u_{in}) \leq b_j, \; j = 1, \dots, J, n = 1, \dots, N \tag{8}$$
$$\textstyle\sum_{i=i}^{I} z_i \leq C \tag{9}$$
$$\textstyle\sum_{n=1}^{N} u_{in} + z_i \leq y_i, i = 1, \dots, I \tag{10}$$

where C is daily capacity.

Transition Probabilities:

$$(x_{11}, \dots, x_{1N}, \dots, x_{I1}, \dots, x_{IN}, y_1, y_2, \dots, y_I) \\ \rightarrow \; (x_{12} + u_{12}, \dots, x_{1N} + u_{1N}, 0, x_{22} + u_{22}, \dots, x_{2N} + u_{2N}, 0, \dots, x_{I2} + u_{I2}, \dots, x_{IN} + u_{IN}, 0, y'_1 \\ + y_1 - \sum_{n=1}^{N} u_{1n} - z_1, y'_2 + y_2 - \sum_{n=1}^{N} u_{2n} - z_2, \dots, y'_I + y_I - \sum_{n=1}^{N} u_{In} - z_I)$$

Costs: $c(u, z) = \sum_{i=1}^{I} \sum_{n=1}^{N} C^i(n, D^i) + \sum_{i=1}^{I} d(i) z_i + w(i)(y_i - \sum_{n=1}^{N} u_{in} - z_i)$, where D^i is a deadline associated with type-i job, $C^i(n, D^i)$ is delay cost of scheduling type-i job on day n, $d(i)$ is rejection cost of type-i job, and $w(i)$ is holding cost for type-i job. $C^i(n, D^i)$ for $i = 1, \dots, I$ is expressed as

$$C^i(n, D^i) = \max(n - D^i, 0) \times F^i, n = 1, \dots, N, \tag{11}$$

where F^i for $i = 1, \dots, I$ is unit delay cost.

Bellman's Equations:

$$v(x, y) = \min_{(u,z)} \{c(u, z) + \lambda \textstyle\sum_{y' \in D} (y') v(x', y')\}, \tag{12}$$

where D is the set of demand vectors. It is however computationally intractable to solve our MDP model due to extremely huge number of states and actions. We therefore apply a direct search-based approximate dynamic programming for the task of approximately solving our problem.

4.1.2 Direct search-based ADP

We use the basis functions of the following type throughout the implementation of the ADP centered on direct-search.

$$\Phi_1(s) = \sum_{n=1}^{N} \sum_{i=1}^{I} x_{in} a_{i1}$$
$$\Phi_2(s) = -(\sum_{i=1}^{I} z_i)$$

The first basis function represents available capacity whereas the latter enables to try different values of z_i for the respective optimization problem.

4.2 Numerical Results

We primarily used the data generation method explained in Göçgün & Ghate (2012). Number of job types was set to two levels: 5 and 10. The length of the booking horizon was set to 14. We set $J \in \{1,2\}$, and $a_{ij} = i + \lceil 2 \times i \times U(0,1) \rceil$ for the case with multiple resource constraints. b_j was set to $t.r \times \sum_{i=1}^{I} a_{ij}$, where $t.r$ is called tightness ratio (Göçgün & Ghate, 2012). $t.r$ was set to $\{1,2\}$. $D(i)$ was set to $DU(1,N)$ for each i. We set $F^i = DU(1{,}100)$ and $d(i) = DU(1{,}100)$ for each i. Holding cost for type-i job was set to $DU(1{,}100)$. λ was set to 0.90 and 0.99.
We present results in Tables 3 and 4. When there is no waiting list in the system, average percentage improvement obtained by the direct search-based ADP turns out to be 22% (see Table 3). Whereas when there is waiting list, the percentage improvement is around 76% (see Table 4).

Table 3. Results for advanced scheduling with multiple resources and no waiting list.

$(I, \lambda, t.r)$	Myopic	DS-ADP	Per. impr.
(5,0.99,1)	19307	16057	16.8
(5,0.99,2)	11254	7844	30.3
(5,0.9,1)	3356	2783	17.1
(5,0.9,2)	1347	1089	19.1
(10,0.99,1)	41752	33629	19.4
(10,0.99,2)	7133	5849	18
(10,0.9,1)	283150	72073	28.3
(10,0.9,2)	2607	1901	27

Table 4. Results for advanced scheduling with multiple resources and waiting list.

$(I, \lambda, t.r)$	Myopic	DS-ADP	Per. impr.
(5,0.99,1)	159988	23183	85.5
(5,0.99,2)	67536	9730	85.5
(5,0.9,1)	13161	3811	71
(5,0.9,2)	4132	1189	71.2
(10,0.99,1)	283150	72073	74.5
(10,0.99,2)	125499	14783	88.2
(10,0.9,1)	23302	9389	59.7
(10,0.9,2)	125499	2748	97.8

4.3. Discussion

Using the abovementioned results, we make the following inferences. The direct-search based ADP technique is able to explore a wide variety of solutions through basis functions utilized by direct search. What is more, when there is no waiting list and number of types is small, increase in tightness ratio results in increase in percentage improvement.

Further, the direct search-based ADP yields more promising results for advanced scheduling problems than allocation scheduling problems. This implies that this type of ADP can be utilized for solving other classes of advanced scheduling problems arising in fields such as healthcare and manufacturing.

5. Conclusions

Dynamic stochastic scheduling (DSS) problems are commonly observed in diverse fields. Because of computational intractability, researchers resort to approximation techniques for solving large-sized DSS problems. In this research, we focused on two classes of DSS problems: allocation scheduling and advanced scheduling. We developed an ADP cenetered on direct-search for approximately solving the MDP models of these problems, and compared its performance against the myopic policy. Numerical experiments revealed that it is worth utilizing direct search for these kinds of problems since the resulting percentage improvement over the myopic policy is significantly high.

It is worth stating that the performance of the direct search-centered ADP is contingent on the basis functions. Future research may address the performance of direct-search based ADPs using various basis functions.

References

Abdırahman, A. H. Direct search-based approximate dynamic programming for dynamic stochastic advanced scheduling. Master's Thesis, Department of Industrial Engineering, Altinbas University, Turkey, 2019.

Adelman D. Price-directed replenishment of subsets: methodology and its application to inventory routing, *Manufacturing and Service Operations Management*; Vol. 5: pp. 348-371, 2003.

Adelman D. A price-directed approach to stochastic inventory routing, *Operations Research*, Vol. 52: pp. 499-514, 2004.

Altman E. Applications of Markov Decision Processes in communication networks, *Handbook of Markov Decision Processes: Methods and Applications*, pp. 489-536, 2002.

Barz, C., Rajaram, K., Elective patient admission and scheduling under multiple resource constraints. *Production and Operations Management*, pp. 1–24, 2015.

Ceschia, S., Schaerf, A., Dynamic patient admission scheduling with operating room constraints, flexible horizons, and patient delays. *Journal of Scheduling,* Vol. 19: 377-389, 2016.

Göçgün, Y., Ghate, A. A Lagrangian approach to dynamic resource allocation. *Proceedings of the Winter Simulation Conference,* pp. 3330-3338, 2010.

Göçgün, Y., Ghate, A. Lagrangian relaxation and constraint generation for allocation and advance scheduling. *Computers Operations Research*, Vol. 39: pp. 2323-2336, 2012;.

Göçgün, Y., Puterman, M.L. Dynamic scheduling with due dates and time windows: an application to chemotherapy patient appointment booking. Health Care Manag Sci, Vol. 7: 60-76, 2014.

Göçgün Y, Dynamic Scheduling with Cancellations: An Application to Chemotherapy Appointment Booking. *An International Journal of Optimization and Control: Theories and Applications*, Vol. 8, no. 2: 161-169, 2018.

Göçgün, Y, Performance comparison of approximate dynamic programming techniques for dynamic stochastic scheduling. *An International Journal of Optimization and Control: Theories & Applications*, Vol. 11*,* no. 2, pp. 178–185, 2021.

Huh, W. T. , Liu, N., Truong, V. A., Multiresource allocation scheduling in dynamic environments. Manufacturing Service Operations Managemen, Vol. 15, no. 2, pp. 280–291, 2013.

Parizi M S, Ghate A. Multi-class, multi-resource advance scheduling with no-shows, cancellations and overbooking. Computers Operations Research, Vol. 67, pp. 90-101, 2016.

Patrick J, Puterman M L, Queyranne M. Dynamic multi-priority patient scheduling for a diagnostic resource. Operations Research, Vol. 56: 1507-1525, 2008.

Powell W B. Approximate Dynamic Programming: Solving the curses of dimensionality of dimensionality. Hoboken, New Jersey, USA: John Wiley and Sons, 2007.

Saure A, Patrick J, Tyldesley S, Puterman M. L. Dynamic multi-appointment patient scheduling for radiation therapy. European Journal of Operational Research, Vol. 223, pp. 573-584, 2012.

Wang X, Truong V, Bank D. Online Advance Admission Scheduling for Services with Customer Preferences. Working Paper, 2015.

Biographies

Yasin Göçgün received his B.S. degree and M.S. degree from the Industrial Engineering Department at Bilkent University in 2003 and 2005, respectively. After completing his doctoral studies in the Industrial and Systems

Engineering Department at the University of Washington in 2010, Dr. Göçgün worked as a postdoctoral fellow in Canada between 2010 and 2014. Prior to joining the Industrial Engineering Department at İstinye University, Dr. Göçgün worked as an assistant professor in the Industrial Engineering Department at Altınbaş University between 2014 and 2020.