İzmir İktisat Dergisi
İzmir Journal of Economics

ISSN:1308-8173 **E-ISSN:** 1308-8505 **YIL:** 2021 **Cilt:** 36 **Sayı:** 2 **Sayfa:** 409-422
Geliş Tarihi: 01.10.2020 **Kabul Tarihi:** 12.07.2021 **Online Yayın:** 05.08.2020 **Doi:** 10.24988/ije.202136211
ÖZGÜN ARAŞTIRMA

Seasonal Reservation Scheduling with Resource Costs: A Mathematical Modeling Approach

Uğur ELİİYİ [1]

Abstract

In this study, a novel optimization problem for simultaneous capacity planning and scheduling in reservation scheduling environments is proposed. The problem is important for seasonal reservation systems such as hotel or seat reservations of travel agencies, or operation and treatment reservations in health tourism. The objective of the problem is to maximize the net profit gained from the processed reservations. To the best of our knowledge, the problem was not previously studied. An integer programming model is developed for exact solutions and extensive computational experimentation reveals model performance under different scenarios. The results are analyzed, and managerial implications are discussed.

Keywords: *Optimization, Reservation Scheduling, Integer Programming, Time Windows, Capacity Planning.*
Jel Codes: *M11, C61, L83.*

Kaynak Maliyetli Dönemsel Rezervasyon Çizelgeleme: Bir Matematiksel Modelleme Yaklaşımı

Özet

Bu çalışmada, rezervasyon çizelgeleme ortamlarında eş zamanlı kapasite planlama ve çizelgeleme için yeni bir optimizasyon problemi önerilmiştir. Problem seyahat acentelerinin otel ve koltuk rezervasyonları veya sağlık turizminde operasyon ve tedavi rezervasyonları gibi dönemsel/sezonluk rezervasyon gerektiren pek çok sistemin optimizasyonu açısından önemlidir. Önerilen problemde işlenen rezervasyonlardan elde edilen net kârın maksimize edilmesi amaçlamaktadır. Kapasite ve çizelgeleme kararlarını içeren bu çizelgeleme problem hizmet endüstrisinde geniş uygulama alanına sahiptir ve bilgimiz dahilinde daha önce incelenmemiştir. Çalışmamızda optimal çözümler için bir tamsayı programlama modeli geliştirilmiş ve kapsamlı sayısal deneylerle farklı senaryolar altında model performansı ölçümlenmiştir. Deney sonuçları analiz edilerek yönetimsel etkileri tartışılmıştır.

Anahtar Kelimeler: *Optimizasyon, Rezervasyon Çizelgeleme, Tamsayılı Programlama, Zaman Aralıkları, Kapasite Planlama.*
Jel Kodu: *M11, C61, L83.*

1. INTRODUCTION

Scheduling problems in optimization have a large variety, including various task characteristics (precedence relations, preemption, ready times, time windows, etc.), resource characteristics (identical, uniform, unrelated) and several objectives (makespan, flow time, lateness or tardiness, etc.). It is also a well-studied area of optimization due to its practical importance and large application areas. While the decision maker has the liberty of determining the start times of tasks in conventional scheduling, these start times are defined as parameters in Interval Scheduling (IS). Besides its evident practical importance in various manufacturing systems such as maintenance scheduling and shift scheduling, the IS problem has important applications in reservation systems, which can be defined as systems in which reservation requests are collected and processed for allocating the available times of the resources. Some examples to reservation systems include airline seat reservation, operation room scheduling,

ATIF ÖNERİSİ (APA): Eliiyi, Uğur. (2021). Seasonal Reservation Scheduling with Resource Costs: A Mathematical Modeling Approach. İzmir İktisat Dergisi. 36(2). 409-422. Doi: 10.24988/ije.202136211

[1] Asst. Prof. Dr., İzmir Bakırçay University, Faculty of Economics and Administrative Sciences, Seyrek, Menemen / İZMİR, **EMAIL:** ugur.eliiyi@bakircay.edu.tr **ORCID:** 0000-0002-5584-891X

classroom scheduling, hotel room reservations, etc.

The decision maker faces two decisions in an IS problem; namely, whether to process an incoming task (customer, job, reservation etc.) and which resource (server, machine, hotel room, seat, rental car etc.) to assign to the task if it will be processed. In a typical IS environment, m resources/processors are available for serving n incoming reservation requests / tasks. The start of the time window of task j is specified by its ready time r_j, whereas the end of the time window is defined by its deadline d_j.

The ready time and deadline of a task identify its reservation interval. A tactical IS problem minimizes the total cost $\sum c_k$ of the resources necessary to serve all tasks at hand. In other words, this problem answers the question of "What is the optimal number/cost of resources required to satisfy all incoming demand?". On the other hand, the operational problem takes the number of resources as a given parameter of the optimization problem, and finds a subset of tasks to serve, so that the resulting total profit $\sum w_j$ from the served tasks is maximized.

The IS problem has two classes in literature. If the tasks/jobs cannot be delayed to start their processing after their ready times, the resulting IS problem is named as the Fixed Job Scheduling (FJS) problem. On the other hand, if the problem includes a time window for each task in which it can be started, this generalization of the FJS is called the Variable Job Scheduling (VJS) problem, or parallel machine scheduling with time windows (Gabrel, 1995). In VJS, each task entering the system at its ready time r_j should start its processing latest on its standby limit $b_j(> r_j)$, otherwise it will be lost. Since VJS is a sub-class of the IS problem, tactical and operational versions of VJS (TVJS and OVJS) are defined similarly. These two versions are usually handled individually in literature.

To the best of our knowledge, capacity planning and scheduling in VJS have not been studied together. In this study, we combine the properties of the tactical and operational aspects of the problem to introduce a novel model. The Combined Reservation Scheduling (CRS) model proposed in this study integrates these decisions, assuming profits for tasks and fixed resource costs. The objective function of the problem is profit maximization for the served subset of tasks after the resource costs are deducted.

While the proposed CRS problem can be used to determine the capacity and schedule of a reservation system, it can also be utilized for finding the number additional resources to be added into an existing resource pool, as well as the resulting optimal schedule. Therefore, repeated solutions of the CRS problem through seasons can be useful in reservation systems showing demand seasonality.

1.1 Motivation

The problem defined in this study is relevant and applicable to many practical situations. We are primarily motivated from the real-life problem of a travel agency working with many different hotel chains throughout the season and bringing tourists from various countries via bookings. The reservations are usually made in the off-season, and a good deal of hotel rooms are pre-reserved by the agency in many different hotels. Once taken, the details of each reservation request are known and deterministic. However, the reservations are due to changes and cancellations. In addition, many new reservation requests can be made as the season approaches.

In such a volatile and dynamic environment, the agency needs to determine how many more hotels and hotel rooms should be added to its repertoire for taking additional reservations, as well as how many of the new reservation requests should be met. In such a case, although the hotel rooms have similar capacities, their costs can have a wide range. Also, different profit levels can be obtained from tourist groups coming from different countries. Within the scope of this problem, the tasks may refer to

individual guests as well as tours containing groups of people.

1.2 Contribution

As to the best of our knowledge, the CRS problem defined in this study has not been previously studied in literature. The problem can be applied to many other reservation systems such as health tourism (physical therapy, thermal therapy, ocular surgery operations, cosmetic surgery operations etc.), airline reservations, tour/cruise bookings, as well as its many applications in the manufacturing environments. As another application area for this problem, the airport gate assignment problem for passenger planes can be considered. In such a problem environment, the number of gates to be used (out of all available gates with different usage costs) and the schedule on each gate can be determined by solving the proposed problem in this paper.

The rest of the paper is structured as follows. In the next section, related studies are reviewed. In Section 3, the CRS problem is defined, formulated, and explained in detail. Results of the computational experimentation for evaluating the performance of the developed model are provided in Section 4, along with discussions and managerial implications. Finally, conclusions and future research ideas are elaborated in Section 5.

2. RELATED WORKS

The IS problems have several application areas in production and service environments. Kroon (1990), and Kolen and Kroon (1992) focused on the airport gate assignment problem and maintenance workforce capacity planning via the tactical and operational FJS models. Fischetti et al. (1987, 1989, 1992) modeled the bus driver scheduling problem through a tactical FJS. Kolen and Kroon (1991, 1993) used the FJS problem in the context of classroom scheduling, also using availability constraints for the resources. Even satellite scheduling was modeled and solved as an operational FJS problem (Wolfe and Sorensen, 2000). Circuit board printing (Spieksma, 1999) and data packet transfers (Faigle et al., 1999) are other interesting applications of the problem. Some variations involve eligibility constraints (Kroon, 1990; Eliiyi and Azizoglu, 2009), operating time restrictions on the resources (Fischetti et al., 1992; Eliiyi and Azizoglu, 2011), or different resource processing speeds (Azizoglu and Bekki, 2008). Eliiyi (2013) defined problems to handle concurrent capacity and scheduling decisions in FJS. The study also includes working time constraints for the resources. As well as mathematical models, heuristic algorithms were also presented with their worst-case time complexities. Kovalyov et al. (2007) and Kolen et al. (2007) reviewed the literature for the FJS problem together with its application areas and complexity results.

While the IS problems have been extensively considered as FJS in literature, studies on the VJS problem are scarce after the original study by Gertsbakh and Stern (1978). These authors formulated the basic TVJS problem with identical resources and proposed an approximate solution for the problem. Gabrel (1995) used the operational FJS model in the context of satellite data transfer scheduling and stated that their model and solution approach could also be modified for solving OVJS. The author dealt with the number-maximizing operational FJS problem with identical task weights and eligibility considerations. Lower and upper bounds for the problem are developed and the computational results are presented. Extension to the OVJS problem is discussed but no computational study is reported.

Very few studies focused on OVJS in literature. Rojanasoonthon et al. (2003), Rojanasoonthon and Bard (2005), and Bard and Rojanasoonthon (2006) all considered the problem for a data relay satellite system and provided many interesting application areas. In their studies, the tasks have different classes and multiple time windows. Task priorities were defined such that the high priority tasks

were infinitely more important than the low priority tasks. The authors used a vehicle-routing-type formulation to accommodate the sequence-dependent setup requirements (Rojanasoonthon et al., 2003).

A dynamic programming-based heuristic and a greedy randomized adaptive search procedure (GRASP) was proposed for the problem (Rojanasoonthon et al., 2003; Rojanasoonthon and Bard, 2005). A branch and price algorithm for the same problem was developed by Bard and Rojanasoonthon (2006), and the computational results revealed that the developed method was able to solve many moderately large instances to optimality.

Garcia and Lozano (2005) studied OVJS problem with two stages in a ready-mix concrete manufacturing context. Their problem resembles a no-wait flowshop with parallel resources and time windows. An ideal start time was defined for each task, and the objective minimized the total weighted deviation from the ideal start times as well as maximizing the weight sum of the selected tasks. They proposed a tabu search heuristic to solve the problem, and the results indicated a good performance in terms of time and quality of solutions.

Eliiyi et al. (2009) handled the OVJS problem in the context of berth allocation in a container port. In their study, two task classes represented two different sizes for vessels, and the two resource classes corresponded to small and large berths for assigning these vessels. They proposed an integer programming model and proved the problem to be NP-hard. An algorithm was designed for generating good initial solutions based on constraint-graphs. Improvement algorithms including genetic algorithm were also developed. The results indicate a superior performance of the developed heuristic as compared to the genetic algorithm.

The number of resources in a reservation system is the most important factor governing the potential profit, as it finds the set of tasks that can be served. In this respect, the capacity plan of a reservation system should be made cautiously. Although previous studies use the TVJS models for capacity planning in VJS environments, the tactical problem necessitates deterministic arrivals or long-term forecasts of reservations, which may be subject to changes, and it ignores possible cancellations or new requests during this long planning period.

Consequently, validity and applicability issues arise while using TVJS in capacity planning. Moreover, capacity planning via TVJS ignores the capacity change requirements, which is an important issue where seasonal demand fluctuations are of concern. The novel model developed in this study, which will be presented in the next section addresses and resolves these drawbacks.

3. THE COMBINED RESERVATION SCHEDULING PROBLEM

In the reservation problem handled in this study, n tasks (reservations) are to be served on the available resources. The upper bound on the number of available resources (e.g., available hotel rooms to be rented by the travel agency for the whole season) is denoted by m. This number could be limited externally by resource availability, or it may be set to a sufficiently large value if the number of available resources is unlimited, as will be explained below.

Each resource has an associated fixed cost, c_k, which may denote its rental fee to the agency for the whole season. Each task j has a ready time denoted by r_j and a standby time standby limit $b_j(> r_j)$, till which it can wait to be served. If the task is delayed/unserved past its standby time, it is considered as lost. The standby period corresponds to a customer's specifying specific dates for the hotel reservation and indicating $\pm x$ days of flexibility for this reservation. The processing time (reservation duration) and the profit from task j are denoted by p_j and w_j, respectively.

The following assumptions are valid throughout the rest of the paper: All problem

parameters r_j, b_j, p_j, w_j, and c_k are nonnegative integers and deterministic. The deterministic parameter assumption is based on the following practical fact: Once a reservation request is taken, the start and the duration of the reservation, as well as the standby limit and expected profit are immediately determined and fixed. If a reservation request is cancelled or altered before it is processed, this can be taken as a new reservation request. Due to this observation in real-life systems, the parameters of the problem can be taken as deterministic and updated as necessary.

Another approach in literature for coping with reservation/demand fluctuations is to consider each reservation request individually and in an online fashion, which results in taking perpetual screenshots of the reservation system for each incoming request and taking decisions dynamically over time. This approach has been studied especially in the communications network context where changes occur very frequently and the planning horizons are relatively shorter. (Steiger et al., 2004; Barshan et al., 2016). In the stochastic variant of online interval scheduling problems, the job parameters are assumed to follow a distribution, whereas in adversarial online interval scheduling there is no such assumption (Yu and Jacobson, 2020).

Another assumption of the defined problem is that, a task can only be served by a single resource without any interruption, i.e. once a customer is assigned a room, that room will not be changed against the customer's will during the entire duration of the reservation.

3.1 Integer Programming Model

In order to be able to formulate a mathematical model for this problem, the planning horizon (the season) is divided into intervals of unit time length, resulting in T time intervals as $\{t_1, \dots, t_T\}$. P_a is defined as the set of tasks available for processing in time interval $[t_a, t_{a+1})$, where $a = 1, \dots, T-1$. That is;

$$P_a = \{j | r_j \leq t_a, b_j + p_j - 1 \geq t_a\}.$$

We also define set S_j as the complete set of intervals for task j, i.e.,

$$S_j = \{r_j, \dots, b_j + p_j - 1\}.$$

The decision variables for the model are defined as follows:

x_{jka}: $\begin{cases} 1, \text{ if task } j \text{ is served on resource } k \text{ in time interval } a \\ 0, \text{ otherwise} \end{cases}$

y_{jk}: $\begin{cases} 1, \text{ if task } j \text{ is served on resource } k \\ 0, \text{ otherwise} \end{cases}$

z_k: $\begin{cases} 1, \text{ if resource } k \text{ is used} \\ 0, \text{ otherwise} \end{cases}$

Decision variable x_{jka} takes the value of 1 if task j is served by resource k in the time interval $[t_a, t_{a+1})$; that is, the a^{th} time interval. Note that this definition might allow a particular task to be served by different resources in different time intervals. This preemption is avoided through the definition of the second decision variable, y_{jk}. This variable takes the value of 1 if all intervals of a particular task j is served on resource k (by means of constraint 2 in the below model). The third decision variable z_k is necessary for calculating the fixed costs of the resources. If a resource is utilized for serving any task, its corresponding z_k variable takes the value of 1, and its fixed cost becomes active in the objective function.

With these definitions and assumptions, the integer programming model for CRS is presented below. The model determines the capacity level of the system (how many resources will be used in total) as well as the processed task subset (how many tasks will be served) simultaneously.

CRS:

$$\text{Maximize} \sum_{j=1}^{n} \sum_{k=1}^{m} w_j y_{jk} - \sum_{k=1}^{m} c_k z_k \quad (1)$$

s.t.

$$\sum_{a \in S_j} x_{jka} = p_j y_{jk}, j = 1, \dots, n; k = 1, \dots, m \quad (2)$$

$$\sum_{k=1}^{m} y_{jk} \leq 1, j = 1, \dots, n \quad (3)$$

$$\sum_{j \in P_a} x_{jka} \leq 1, a = 1, \dots, T-1; k = 1, \dots, m \quad (4)$$

$$p_j x_{jka} - p_j x_{j,k,a+1} + \sum_{t=a+2}^{b_j+p_j-1} x_{jkt} \leq p_j,$$

$$j = 1, \dots, n; k = 1, \dots, m; a \in S_j \quad (5)$$

$$y_{jk} \leq z_k, j = 1, \dots, n; k = 1, \dots, m \quad (6)$$

$$x_{jka}, y_{jk}, z_k \in \{0,1\},$$

$$j = 1, \dots, n; a = 1, \dots, T-1; k = 1, \dots, m \quad (7)$$

The objective function in (1) maximizes the net total profit, defined as the total profit obtained from serving the reservation subset, minus the resource usage/rental fixed costs. Constraints (2) ensure that all intervals of a task are assigned to the same resource. Constraints (3) limit the assignment of a task to at most one resource. Constraints (4) guarantee that each resource can process at most one task in an interval. Constraints (5) assure the continuity of a task's intervals, i.e., if a task is processed, each time interval of that task should be assigned consecutively to the same resource. Constraints (6) ensure that a resource is used if any task is processed on it. Finally, constraints (7) define the binary structure of the decision variables.

3.2 Upper Bound on the Number of Resources

Arkin and Silverberg (1987) solved the tactical FJS problem. The resulting value gives the minimum number of serving resources to handle all tasks in the system, which yields an upper bound for the number of resources for the operational problem. The maximum number of task overlaps is computed as $\text{Max}_a\{|P_a|\}$, where $|P_a|$ is the cardinality of set P_a.

Using these results, the number of resources (m) in the model above may be set to a sufficiently large value by the following upper bounding procedure. First, we pessimistically assume that each task has a deadline equal to its $b_j + p_j$. That is, each task starts at its ready time and ends at its $b_j + p_j$. Then, an upper bound on the number of resources is computed as $m = \text{Max}_a\{|P_a|\}$.

3.3 Computational Complexity

The CRS problem reduces to the OVJS problem when $c_k = 0, \forall k$. The OVJS problem is reported as NP-hard (Gabrel, 1995). Hence, the CRS problem is also NP-hard.

4. EXPERIMENTATION

A computational experiment is conducted to assess the performance of the mathematical model. IBM ILOG CPLEX 12.8 is used for obtaining the optimal solutions. A PC configuration with Core 2 Duo 2.8 GHz, 4 GB memory is used for the experiments. The ready times of the incoming reservations are uniformly generated between 0 and 200, i.e., the season is 200 time units (e.g. days) long and the reservation requests occur uniformly within this interval. Two levels are considered for the processing time parameter corresponding to low and higher variability cases:

- $p = 1$ level: processing time chosen uniformly between 4 and 10 for each task, corresponding to low variability among the processing times.
- $p = 2$ level: processing time chosen uniformly between 4 and 20 for each task, corresponding to higher variability among the processing times.

Three levels are used for task profits:

- $w = 1$ level: Each task's profit is equal to its processing time.
- $w = 2$ level: profit chosen uniformly between 4 and 10 for each task, corresponding to low variability among task profits.
- $w = 3$ level: profit chosen uniformly between 4 and 20 for each task, corresponding to a higher variability among task profits.

Three levels are set for the cost of resources:

- $c = 1$ level: cost of resource chosen uniformly between 80, 100, 120, 140, 160 for each resource.
- $c = 2$ level: equal cost of resources taken as 80, low-cost case.

- $c = 3$ level: equal cost of resources taken as 160, high-cost case.

As it can be observed from the above experimental settings, in $c = 1$ level, the lowest cost for a resource is set as 20 times the lowest profit of tasks. The highest cost is set as twice the lowest due to possible service level difference such as luxury packages, price promotions or other possible reasons.

Two levels are considered for the standby durations of the tasks ($b_j - r_j$):

- $b - r = 1$ level: standby duration chosen uniformly between 0 and 10 for each task, corresponding to low variability.
- $b - r = 2$ level: standby duration chosen uniformly between 0 and 20 for each task, corresponding to higher variability.

The inclusion of zero in the uniform distribution intervals generalizes our experiment to include the FJS problem, since the problem becomes FJS when $b_j - r_j = 0$.

Ten test problems are generated for each of the 36 experimentation scenarios, with $n = 20, 50, 100, 200$ tasks. Here, $n = 20$ corresponds to a small problem and $n = 200$ corresponds to a very large problem, especially when n might represent a group of reservations in some cases. Therefore, a set of 1440 problem instances are generated and solved in total.

4.1 Results and Discussion

We discuss the performance of our model using the outputs of the optimal solutions. A 1200-second time limit is imposed on CPLEX for obtaining the optimal solutions. For the instances that could not be solved to optimality within the predetermined time limit, the best feasible solution obtained by CPLEX is reported. Tables 1 through 3 present the result of the computational experiment at three different resource cost levels.

The column Avg. UB corresponds to the upper bound on the number of resources, calculated as explained in Section 3.2 and averaged over 10 instances. The values in this column represent the necessary number of resources to process all incoming reservation requests. The other columns compare the average number of used resources among the available ones, the average percentage of resource utilization for the used resources, the percentage of processed tasks over all tasks in the system, the percentage of processed profit over the total potential profit from all tasks, and the model solution times from CPLEX.

The *Avg. # used* column represents the number of used resources in the solution, out of the available number of resources indicated by the upper bound. These values are averaged over 10 problem instances of the same experiment setting. For example, in Table 1, for $n = 50$, $b - r = 1$, $w = 1$ and $p = 2$, while all incoming reservation requests in the system could be processed with nine resources, the best solution used only two resources on the average.

The *Avg. % util.* (resource load / length of the season) is the resource utilization percentage during the season $[0, 200]$. For example, in Table 1, for $n = 50$, $b - r = 1$, $w = 1$ and $p = 2$, the solutions used only 47% of the two resources.

The average percentage of processed tasks *(% tasks processed*) computed as $\sum_{j=1}^{n} x_{jk}/n$, and the average percentage of processed profit *(% profit processed*) computed as $\sum_{j=1}^{n} w_j x_{jk} / \sum_{j=1}^{n} w_j$ for the CPLEX solutions constitute the next columns. The percentage of processed profit can be seen as an indicator of how much of the potential gain could be obtained with the optimal solutions. For the same example above, it seems that with the CPLEX solution 41% of the tasks are processed, corresponding to 42% of the potential profit. Note that, for this example, the average percentages of tasks and profits are very close to each other. This is since every task's profit is equal to its processing time at the $w = 1$ level of the profit parameter. The percentages differ for other settings where the processed profit percentage becomes higher than processed task percentage.

Table 1: Results for $c_k \sim U\{80,100,120,140,160\}, \forall k$.

n	*b-r*	*w*	*p*	Avg. UB	Avg. # used	Avg. % util.	% tasks processed	% profit processed	Solution Time (sec.)
20	1	1	1	4	1	45	75	76	4
			2	5	1	60	61	65	159
		2	1	4	1	43	74	77	2
			2	5	1	27	31	32	10
		3	1	4	1	43	75	80	3
			2	5	1	52	62	69	31
	2	1	1	5	1	53	89	89	7
			2	5	1	65	68	70	273
		2	1	4	1	52	91	92	2
			2	5	1	52	55	59	144
		3	1	5	1	50	86	90	4
			2	5	1	63	69	76	50
50	1	1	1	8	2	66	75	75	1200
			2	9	2	47	41	42	1200
		2	1	8	2	66	69	73	1200
			2	9	1	31	19	20	1164
		3	1	7	2	61	83	88	1200
			2	10	2	55	42	46	1200
	2	1	1	10	2	67	67	68	1200
			2	10	1	33	18	18	1200
		2	1	9	1	66	65	69	1200
			2	10	1	31	22	24	1200
		3	1	9	2	62	78	83	1200
			2	11	1	34	26	29	1200
100	1	1	1	12	1	28	16	16	1200
			2	15	0	0	0	0	1200
		2	1	12	1	29	19	21	1200
			2	15	1	3	1	1	1200
		3	1	13	2	33	29	31	1200
			2	14	0	10	2	2	1200
	2	1	1	15	1	39	21	22	1200
			2	19	0	4	1	1	1200
		2	1	16	1	50	26	28	1200
			2	19	1	5	4	4	1201
		3	1	14	2	40	36	39	1200
			2	16	1	5	3	3	1200
200	1	1	1	21	0	0	0	0	1200
			2	21	0	0	0	0	1200
		2	1	22	0	2	1	1	1202
			2	22	0	0	0	0	1202
		3	1	19	3	10	7	7	1200
			2	18	0	0	0	0	1200
	2	1	1	26	0	16	3	3	1203
			2	28	0	0	0	0	1207
		2	1	26	1	4	1	1	1206
			2	28	1	4	1	2	1204
		3	1	26	1	11	3	4	1200
			2	26	0	0	0	0	1200

Table 2: Results for $c_k = 80, \forall k$.

n	*b-r*	*w*	*p*	Avg. UB	Avg. # used	Avg. % util.	% tasks processed	% profit processed	Solution Time (sec.)
20	1	1	1	4	1	49	79	79	3
			2	5	1	57	67	69	220
		2	1	4	1	44	81	83	2
			2	5	1	58	65	70	12
		3	1	4	1	45	80	85	4
			2	4	1	54	63	68	71
	2	1	1	5	1	51	91	91	3
			2	5	1	69	75	75	377
		2	1	5	1	51	90	92	4
			2	5	1	63	71	75	92
		3	1	5	1	52	90	92	5
			2	6	1	62	71	76	335
50	1	1	1	8	2	61	84	84	1200
			2	9	3	58	66	67	1200
		2	1	7	2	61	81	84	1200
			2	9	1	60	46	50	1200
		3	1	8	2	58	83	89	1200
			2	10	3	54	62	68	1200
	2	1	1	9	2	66	73	74	1200
			2	10	1	47	26	27	1200
		2	1	8	2	64	72	76	1200
			2	11	1	36	22	24	1200
		3	1	8	2	63	86	90	1200
			2	10	1	42	33	37	1200
100	1	1	1	12	3	52	61	61	1200
			2	15	1	17	7	8	1200
		2	1	12	3	59	59	62	1200
			2	15	0	17	5	6	1200
		3	1	12	4	49	64	68	1200
			2	15	3	39	22	26	1200
	2	1	1	16	1	37	26	25	1199
			2	19	1	13	7	8	1200
		2	1	15	2	49	43	46	1200
			2	18	0	9	4	4	1200
		3	1	15	3	42	47	52	1200
			2	18	2	28	16	19	1200
200	1	1	1	21	1	35	11	12	1200
			2	27	0	0	0	0	1200
		2	1	19	3	49	32	35	1200
			2	27	1	2	1	1	1200
		3	1	21	2	37	18	21	1200
			2	28	0	0	0	0	1200
	2	1	1	27	0	0	0	0	1202
			2	31	0	0	0	0	1200
		2	1	28	0	6	1	1	1201
			2	31	0	2	1	1	1200
		3	1	27	0	0	0	0	1200
			2	32	0	0	0	0	1200

Table 3: Results for $c_k = 160, \forall k$.

n	*b-r*	*w*	*p*	Avg. UB	Avg. # used	Avg. % util.	% tasks processed	% profit processed	Solution Time (sec.)
20	1	1	1	4	0	0	0	0	0
			2	5	0	7	7	6	5
		2	1	4	0	0	0	0	0
			2	4	0	0	0	0	0
		3	1	4	1	44	78	81	3
			2	5	0	16	19	20	8
	2	1	1	5	0	0	0	0	0
			2	6	1	58	55	56	161
		2	1	4	0	0	0	0	0
			2	6	0	0	0	0	0
		3	1	4	1	54	92	95	3
			2	6	1	50	57	62	72
50	1	1	1	7	1	78	52	54	1200
			2	9	1	81	33	32	1200
		2	1	7	1	72	55	60	1200
			2	9	0	8	5	5	1116
		3	1	7	2	64	69	76	1200
			2	9	1	69	42	48	1200
	2	1	1	9	1	41	28	28	1200
			2	10	0	0	0	0	1200
		2	1	10	1	75	56	62	1200
			2	11	0	0	0	0	1200
		3	1	8	1	68	67	75	1200
			2	11	1	46	25	29	1200
100	1	1	1	13	1	49	17	17	1200
			2	15	0	8	5	4	1200
		2	1	12	1	53	23	26	1200
			2	15	0	0	0	0	1200
		3	1	12	2	57	55	62	1200
			2	15	1	10	7	8	1200
	2	1	1	15	0	0	0	0	1200
			2	18	0	0	0	0	1200
		2	1	16	1	40	17	19	1200
			2	19	0	0	0	0	1200
		3	1	14	2	49	36	42	1200
			2	18	0	4	2	2	1200
200	1	1	1	21	0	7	1	1	1200
			2	27	0	0	0	0	1200
		2	1	20	1	25	11	13	1200
			2	26	1	4	2	2	1201
		3	1	20	2	36	24	27	1200
			2	26	0	0	0	0	1200
	2	1	1	26	0	0	0	0	1200
			2	31	0	0	0	0	1200
		2	1	26	0	0	0	0	1200
			2	32	0	0	0	0	1200
		3	1	26	0	0	0	0	1201
			2	33	0	0	0	0	1200

For $n = 20$, CPLEX found the optimal solutions for all instances. Many of the problem instances with $n = 50$ or more tasks could not be solved optimally by CPLEX within the 1200-second time limit, which is an indicator of the difficulty of the optimization problem. On the other hand, best feasible solutions are obtained for these instances.

It can be observed from Table 2 and Table 3 that the optimal solutions are obtained faster for high resource costs than for lower ones. This result is expected, because more tasks can be processed if the resource costs are low, leading to a higher number of solution alternatives. The same trend is valid for longer standby durations and longer processing times, as the increase in these parameters also result in an increased number of feasible solutions. Varying levels of profits obtained from the reservations do not seem to affect the solution times. However, the solution times increase with increasing number of tasks expectedly, as the problem size grows considerably when the number of tasks is high.

4.2 Managerial Implications

It can be observed by comparing the number of used resources in Tables 2 and 3 that, as the costs of resources increase, the decision of capacity expansion is negatively affected. Conversely, higher resource costs bring higher utilization values. This result is expected from a managerial point of view. If the profits to be obtained from the reservations are predetermined and cannot be changed, more costly resources should be avoided while determining the seasonal capacity. However, if somehow a net profit is to be gained even with the utilization of higher-cost resources, then high utilization rates of these resources would further justify their usage.

While the optimal resource utilization percentages are around 50% in solutions of the small instances, the values decrease down to around 20% for the larger instances. This is partly due to the fact that these instances cannot be solved optimally within the given time limit and best feasible solutions by CPLEX are reported. Many of the solutions for the larger instances include a zero objective function value, and the corresponding resource utilization is therefore also zero. The reason for these "do-nothing" decisions by the model can be explained as follows.

Due to the higher number of overlapping task alternatives in larger instances, a denser packing is awaiting to be served by the available resources. However, as one reservation request is served, as it occupies a time interval in the planning horizon, and other requests within this horizon cannot be served. Since the model has to search for all such alternative schedules for the larger instances to find the optimal solution, and since doing nothing (scheduling no reservation and adding no resource into the system) is a feasible solution, it is reported as the best solution within the given time limit.

Another observation from Tables 2 and 3 is that the "do-nothing" solution is preferred more when the resource costs are high. Especially for instances with higher number of jobs and lower profits, the optimal decision of adding no extra capacity to the available resource pool (hence using no resources and serving no reservation requests) is economically justified.

Due to the tactical nature of the CRS problem involving the decisions of capacity increase or expansion, it can be thought that the decision makers might be willing to tolerate relatively longer processing times. Based on this reasoning, one can argue that longer times should be allowed for CPLEX to find the optimal solution. However, further experimentation revealed that even 1-hour and 2-hour limits did not change the results obtained by CPLEX for the large instances of the problem. As stated before, this is evidence to the highly combinatorial nature and the NP-hardness of the problem structure. This issue can also be seen as an indication for the need of heuristic solutions for the problem.

The model developed in this study brings a deterministic solution to a problem, which can present itself in a dynamic environment. In a

problem environment where the demand and cancellations occur instantaneously and are to be priced dynamically, the optimization approaches in literature typically offer policy-based solutions for coping with the dynamic nature of the problem. This is done mainly in two stages, where demand forecasting is done in the first stage and a trade-off between cost and pricing is considered in the second stage to establish a capacity-demand balance. Although this approach can yield more realistic parameter values for the problem, it requires the use of highly nonlinear models and the tractability decreases drastically. Due to this reason, online and rule-based approaches are used more often. The assumption of deterministic parameters in our model provides tractability and ease of use. In a problem environment where the demand and cancellations occur instantaneously and are to be priced dynamically, repetitive solutions of the proposed model will be necessary. Therefore, fast and high-quality solutions can be of great use from a practical perspective. An efficient and effective decision as to how much to expand capacity in a given period, and how to schedule the incoming requests on the expanded resource pool can be very valuable in this respect.

5. CONCLUSION

In this study, we introduce the Combined Reservation Scheduling (CRS) problem for determining the capacity and the schedule of a reservation system simultaneously. The motivation of the problem is defined, many application areas are identified, and the related literature is reviewed.

An integer programming formulation is proposed for the problem, and the problem is shown to be NP-hard. We evaluate the performance of the developed model through extensive computational experimentation. It is observed that the computational times for the optimal solutions increase with growing number of tasks in the system. On the other hand, optimal solution performance does not seem to deteriorate as much with growing number of resources.

As to the best of our knowledge, the CRS problem has not been previously studied in literature. For this reason, the model developed in this study is intended to initiate further studies. The computational results reveal that the problem is very difficult to optimize and obtaining solutions for large instances of the problem take too long to be practical for the decision makers. Therefore, some fast and effective heuristic approaches that exploit the structural characteristics of the problem are required to obtain good and efficient solutions for the large instances.

In addition, focusing on metaheuristic approaches and nature-inspired heuristics for the problem may prove to be worthy, as the problem has a high practical importance. Obtaining high-quality solutions to large instances of this problem can be very valuable for the decision makers in relevant sectors such as tourism, transportation and logistics, health logistics, and manufacturing.

REFERENCES

Arkin, A.M., & Silverberg, E.L. (1987). Scheduling Jobs with Fixed Start and End Times. Discrete Applied Mathematics, 18(1), 1–8.

Azizoglu, M., & Bekki, B. (2008). Operational fixed interval scheduling problem on uniform parallel machines. International Journal of Production Economics, 112(2), 756–768.

Bard, J.F., & Rojanasoonthon, S. (2006). A Branch-and-Price Algorithm for Parallel Machine Scheduling with Time Windows and Job Priorities. Naval Research Logistics, 53(1), 24–44.

Barshan, M., Moens, H., Famaey, J., & De Turck, F. (2016). Deadline-aware advance reservation scheduling algorithms for media

production networks. Computer Communications, 77(1), 26–40.

Eliiyi, D.T. (2013). Integrating tactical and operational decisions in fixed job scheduling. Engineering Optimization, 45(12), 1449–1467.

Eliiyi, D.T., & Azizoglu, M. (2009). A Fixed Job Scheduling Problem with Machine-Dependent Job Weights. International Journal of Production Research, 47(9), 2231–2256.

Eliiyi, D.T., & Azizoglu, M. (2011). Heuristics for operational fixed job scheduling problems with working and spread time constraints. International Journal of Production Economics, 132(1), 107–121.

Eliiyi, D.T., Korkmaz, A.G., & Cicek, A.E. (2009). Operational Variable Job Scheduling with Eligibility Constraints: A Randomized Constraint-Graph-Based Approach. Technological and Economic Development of Economy, 15(2), 245–266.

Faigle, U., Kern, W., & Nawijn, W.M. (1999). A greedy online algorithm for the k-track assignment problem. Journal of Algorithms, 31(1), 196–210.

Fischetti, M., Martello, S., & Toth, P. (1987). The fixed job schedule problem with spread-time constraints. Operations Research, 35(6), 849–858.

Fischetti, M., Martello, S., & Toth, P. (1989). The Fixed Job Schedule Problem with Working-Time Constraints. Operations Research, 37(3), 395–403.

Fischetti, M., Martello, S., & Toth, P. (1992). Approximation Algorithms for Fixed Job Schedule Problems. Operations Research, 40(S1), S96–S108.

Gabrel, V. (1995). Scheduling jobs within time windows on identical parallel machines. European Journal of Operational Research, 83(2), 320–329.

Garcia, J.M., & Lozano, S. (2005). Production and delivery scheduling problem with time windows. Computers & Industrial Engineering, 48(4), 733–742.

Gertsbakh, I., & Stern, H.I. (1978). Minimal resources for fixed and variable job schedules. Operations Research, 26(1), 68–85.

Kolen, A.J.W., & Kroon, L.G. (1991). On the Computational Complexity of (Maximum) Class Scheduling. European Journal of Operational Research, 54(1), 23–38.

Kolen, A.J.W., & Kroon, L.G. (1992). License Class Design: Complexity and Algorithms. European Journal of Operational Research, 63(3), 432–444.

Kolen, A.J.W., & Kroon, L.G. (1993). On the Computational Complexity of (Maximum) Shift Class Scheduling. European Journal of Operational Research, 64(1), 138–151.

Kolen, A.J.W., Lenstra, J.K., Papadimitriou, C.H., & Spieksma, F.C.R. (2007). Interval scheduling: A survey. Naval Research Logistics, 54(5), 530–543.

Kovalyov, M.Y., Ng, C.T., & Cheng, T.C.E. (2007). Fixed interval scheduling: Models, applications, computational complexity and algorithms. European Journal of Operational Research, 178(2), 331–342.

Kroon, L.G. (1990). Job scheduling and capacity planning in aircraft maintenance, Ph.D. Thesis, Rotterdam School of Management, Erasmus University, The Netherlands.

Rojanasoonthon, S., Bard, J.F., & Reddy, S.D. (2003). Algorithms for parallel machine scheduling: a case study of the tracking and data relay satellite system. Journal of the Operational Research Society, 54(8), 806–821.

Rojanasoonthon, S., & Bard, J.F. (2005). A GRASP for parallel machine scheduling with time windows. INFORMS Journal on Computing, 17(1), 32–51.

Spieksma, F.C.R. (1999). On the approximability of an interval scheduling problem. Journal of Scheduling, 2(5), 215–227.

Steiger, C., Walder, H. & Platzner, M. (2004). Operating systems for reconfigurable

embedded platforms: online scheduling of real-time tasks. IEEE Transactions on Computers, 53(11), 1393–1407.

Wolfe, W.J., & S.E. Sorensen (2000). Three scheduling algorithms applied to the earth observing systems domain. Management Science, 46(1), 148–168.

Yu, G., & Jacobson, S.H. (2020). Primal-dual analysis for online interval scheduling problems. Journal of Global Optimization, 77, 575–602.