Proceedings of the 2014 Industrial and Systems Engineering Research Conference
Y. Guan and H. Liao, eds.

Inconsistent Objectives in Operating Room Scheduling

Wei Li[†*]
Victoria L. Mitchell[‡]
Barrie R. Nault[‡]

[†] Mechanical Engineering, University of Kentucky, Lexington, KY 40506
[‡] Haskayne School of Business, University of Calgary, Calgary, AB T2N 1N4
[*] wei.mike.li@uky.edu

Abstract

The efficient scheduling of operating room (OR) slates is critical in a hospital setting, especially given OR slates are linked with preoperative and postoperative services. Wait list minimization is a common goal in OR scheduling, one that is broadly defined and can be achieved from many perspectives. Due to the complexity of OR scheduling, and the absence of industrial engineering techniques in healthcare environments, simple rules are often used in scheduling OR slates - typically, the shortest processing time (SPT) rule and the longest processing time (LPT) rule. The SPT and LPT rules, however, have conflicting outcomes in sequencing surgeries. The application of these rules causes confusion in scheduling OR slates. In this paper, we examine wait list minimization from an operations research perspective, explain why SPT and LPT rules are used to achieve this goal, and discuss their conflicting effect on scheduling performance. We also propose a state space with head, body, and tail (SS-HBT) heuristic as an alternative means to achieve this goal. Through case studies, we show that when applied to scheduling OR slates, our SS-HBT heuristic outperforms two well-established heuristics for wait list minimization.

Keywords
Healthcare Systems, Industrial Engineering, Operations Research, Production Scheduling

1. Introduction

The efficient scheduling of operating room (OR) slates is critical in a hospital setting. The OR slate drives the utilization of preoperative units [1], and post-anesthesia care units (PACU) [2], in addition to the surgical suites. Consequently OR scheduling affects the overall performance of a hospital [3]. Last in a sequence of OR scheduling activities, the OR slate organizes a perioperative (peri-op) process that encompasses preoperative (pre-op), intraoperative, and postoperative (post-op) phases of the surgical sequence. Prior research indicates, OR scheduling methods that fail to take preoperative and postoperative units into account are inefficient [3, 4], leading to problems with resource deployment throughout the perioperative process.

Simple rules are often used in scheduling OR slates. In the absence of industrial engineering techniques, the complexity characterizing health care systems is difficult to model. Consequently, simple rules provide a means to sequence surgical procedures that are straightforward and easy to understand. The most popular rules used in scheduling OR slates are the longest processing time (LPT) rule and the shortest processing time (SPT) rule. These two rules are conflicting in nature, offering very different outcomes in surgical sequencing. In general, the LPT rule can achieve high OR utilization, with low levels of staff overtime, but a large number of delayed cases; whereas the SPT rule can achieve low OR utilization, with high levels of staff overtime, and a small number of delayed cases [3-7]. The aforementioned outcomes of the LPT and SPT rules are applicable only to operating rooms (ORs). To date, there is sparse research on the integration of OR slates with other phases of the perioperative process [5]. In practice, the SPT rule is recommended when ORs are linked with pre-op or post-op phases, e.g., Testi et al. link ORs with admission [1], Marcon and Dexter link ORs with post anesthesia care units (PACU) [8]. Moreover, Gul et al. compare the SPT rule, the LPT rule, and a genetic algorithm (GA) based on minimization of surgical suite overtime and patient waiting time. They recommend the SPT rule over the LPT rule and even over the GA [9]. It is no wonder the conflicting OR outcomes generated by the LPT rule and the SPT rule causes confusion in practice. Hospitals that simply focus on high OR utilization tend to use the LPT rule in scheduling OR slates.

Wait list minimization is a broadly defined goal in OR scheduling. There are two specific objectives that can be applied to wait list minimization from the production scheduling perspective. One is to minimize the maximum completion time, to min (C_{max}), and the other is to minimize the average completion time, to min ($\Sigma C_j/n$), which is the same as minimizing the total completion time if the number of surgeries (n) is fixed, i.e., to min (ΣC_j), where $j = 1, \ldots, n$. If there are a number of n surgeries on the wait list, min (C_{max}) is to minimize the completion time of the last surgery, and min ($\Sigma C_j/n$) is to minimize the average completion time of all surgeries. In production scheduling, the LPT rule can achieve high utilization and the SPT rule can minimize flow time [10]. Utilization relates to the maximum completion time, i.e., finishing all jobs in a short period of time, and flow time relates to the average completion time, i.e., every job flows out of a production line fast. Obviously, either objective can be used to minimize the wait list in OR scheduling. If LPT and SPT rules are conflicting in sequencing surgeries, and relate to min (C_{max}) and min (ΣC_j) respectively, then the question remains as to which objective is most advantageous for OR scheduling.

We analyze the inconsistency between min (C_{max}) and min (ΣC_j), and propose a more efficient method for scheduling OR slates. Our method utilizes a *state space with head, body, and tail* (SS-HBT) heuristic to achieve wait list minimization with OR slates. Through case studies, we show that our SS-HBT heuristic outperforms two well-established production scheduling heuristics, the NEH [11] and CDS [12] heuristics, in achieving this goal.

The remainder of our paper is organized as follows. We give a brief literature review on production scheduling in Section 2, which helps explain why there is little integration of industrial engineering technologies with healthcare systems. We analyze the two objectives, min (C_{max}) and min (ΣC_j), in Section 3, to show why they are inconsistent with each other. We propose a new method for scheduling OR slates, applying our SS-HBT heuristic in Section 4. We give results of case studies in Section 5, draw conclusions and discuss future work in Section 6.

2. Literature Review

OR slates can be modeled as a 3-machine flow shop, where machine 1 represents input (pre-op) units, machine 2 represents ORs (intraoperative units), and machine 3 represents output (post-op) units. The literature on flow shop production scheduling is vast. We streamline this knowledge pool, highlighting the most relevant literature addressing this context.

Research on flow shop production scheduling has a long history, dating back to the 1950s [13]. During the first two decades (1955-1974), the research on flow shop production scheduling focused on seeking optimal solutions by combinatorial approaches and branch-and-bound methods. However, Garey et al. in 1976 demonstrated that to min (C_{max}) for n-job m-machine flow shop production is NP-complete [14], where $m \geq 3$ and NP stands for non-deterministic polynomial. Such NP-completeness theory [15] in the third decade (1975-1984) profoundly influenced the research direction of flow shop production scheduling, changing it from seeking optimal solutions by optimization techniques to seeking feasible solutions by heuristics. The time period from 1985 to present witnessed the proliferation of various flow shop techniques, objective functions, and solutions approaches. Research on hybrid flow shop production emerged in the fourth decade (1985-1994). Whereas there is only one machine at each workstation in a traditional flow shop, a hybrid flow shop consists of m workstations with more than one machine at each workstation. Min (ΣC_j) was proposed in the fifth decade (1995-2004) and demonstrated in 1999 by Hoogenveen and Kawaguchi to be NP-complete for 2-machine flow shop production [16].

According to Ruiz and Maroto [17], the NEH heuristic is the best among 19 constructive heuristics, and the CDS heuristic is the eighth for traditional flow shop production to min (C_{max}). However, for hybrid flow shop production to min (C_{max}), the CDS heuristic is the best among 6 constructive heuristics, and better than the NEH heuristic [18].

3. The Inconsistency of Two Objectives

For 3-machine flow shop production, to min (C_{max}) is NP-complete, and for 2-machine flow shop production, to min (ΣC_j) is NP-complete. Therefore our analysis of the two objectives is on a 2-machine flow shop to demonstrate the inconsistency.

For n-job 2-machine flow shop production, $C_{i,j}$ is the completion time of each job $j = 1, \ldots, n$ on each machine $i = 1$, 2 (Figure 1). If $p_{i,j}$ is the processing time of job j on machine i, the sum of idle times on machine 2 can be expressed by $\sum_{j=1}^{n} X_{2,j} = \max_{1 \le j \le n}\{K_{2,j}\}$, where $K_{2,j} = \sum_{k=1}^{j} p_{1,k} - \sum_{k=1}^{j-1} p_{2,k}$.

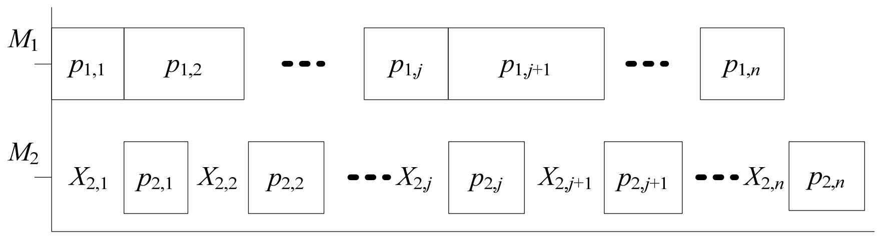

Figure 1: 2-machine flow shop production

For the completion time of each job on machine 2, we have

$C_{2,1} = X_{2,1} + p_{2,1}$
$C_{2,2} = X_{2,1} + X_{2,2} + p_{2,1} + p_{2,2}$
$C_{2,3} = X_{2,1} + X_{2,2} + X_{2,3} + p_{2,1} + p_{2,2} + p_{2,3}$
…
$C_{2,j} = \sum_{k=1}^{j} X_{2,k} + \sum_{k=1}^{j} p_{2,k}$
…
$C_{2,n} = \sum_{k=1}^{n} X_{2,k} + \sum_{k=1}^{n} p_{2,k}$

Therefore, the total completion time is

$$\sum_{j=1}^{n} C_{2,j} = \sum_{j=1}^{n} (n-j+1) X_{2,j} + \sum_{j=1}^{n} (n-j+1) p_{2,j}, \tag{1}$$

which consists of the sum of idle times and the sum of processing times. Each term of $X_{2,j}$ and $p_{2,j}$ will appear $n - j + 1$ times in Equation (1) for $j = 1, \ldots, n$. Define a sequence $S = \{HBT\}$, where $H \cup B \cup T = \{n\}$, and H represents jobs in the head (front) of a sequence, B represents jobs in the body of a sequence, and T represents jobs in the tail (back) of a sequence. One unit of idle time caused by a job in the head of a sequence will have more impact on total completion time, ΣC_j, than that caused by a job in the tail of a sequence. This is the same for processing times, that is, one unit of processing time of a job in the head of a sequence will have more impact on total completion time, ΣC_j, than that of a job in the tail of a sequence.

Given two sequences, $S = \{Hab\}$ and $S' = \{Hba\}$, the impact on total completion time, ΣC_j, by exchanging the positions of the last two jobs a and b, comes from two sources: processing times and idle times. First, we analyze the impact on total completion time from the source of processing times, that is $\sum_{j=1}^{n}(n-j+1)p_{2,j}$. For sequence S in which job a is in position $n - 1$ and job b in position n, we have $A = 2 \times p_{2,a} + p_{2,b}$, and similarly for sequence S', we have $B = 2 \times p_{2,b} + p_{2,a}$. To min ($\Sigma C_j$), we should process job a earlier than job b as long as $A - B \le 0$ from the processing time perspective, and we derive a condition of $p_{2,a} - p_{2,b} \le 0$.

Second, we analyze the impact on total completion time from the source of idle times, that is $\sum_{j=1}^{n}(n-j+1)X_{2,j}$. Comparing the same two sequences, $S = \{Hab\}$ and $S' = \{Hba\}$, and assuming the last job in H causes an idle time, we need to process job a earlier than job b if and only if $\max\{2X_{2,a}, X_{2,b}\} \le \max\{2X'_{2,b}, X'_{2,a}\}$. Thus we compare

$$\begin{aligned}
&\max\{2 \times [(\textstyle\sum_{k=1}^{n-2} p_{1,k} + p_{1,a} - \sum_{k=1}^{n-2} p_{2,k}) - (\sum_{k=1}^{n-2} p_{1,k} - \sum_{k=1}^{n-3} p_{2,k})],\\
&\quad (\textstyle\sum_{k=1}^{n-2} p_{1,k} + p_{1,a} + p_{1,b} - \sum_{k=1}^{n-2} p_{2,k} - p_{2,a}) - (\sum_{k=1}^{n-2} p_{1,k} + p_{1,a} - \sum_{k=1}^{n-2} p_{2,k})\} \le\\
&\max\{2 \times [(\textstyle\sum_{k=1}^{n-2} p_{1,k} + p_{1,b} - \sum_{k=1}^{n-2} p_{2,k}) - (\sum_{k=1}^{n-2} p_{1,k} - \sum_{k=1}^{n-3} p_{2,k})],\\
&\quad (\textstyle\sum_{k=1}^{n-2} p_{1,k} + p_{1,b} + p_{1,a} - \sum_{k=1}^{n-2} p_{2,k} - p_{2,b}) - (\sum_{k=1}^{n-2} p_{1,k} + p_{1,b} - \sum_{k=1}^{n-2} p_{2,k})\}.
\end{aligned} \tag{2}$$

If the condition of max $\{2X_{2,a}, X_{2,b}\} \leq$ max $\{2X'_{2,b}, X'_{2,a}\}$ holds, the condition of max $\{2X_{2,a}, X_{2,b}\}$ – max $\{2X'_{2,b}, X'_{2,a}\} \leq 0$ holds. By max $\{A, B\}$ – max $\{C, D\} \leq$ max $\{A - C, B - D\}$, condition (2) can be expressed as max $\{2 \times (p_{1,a} - p_{1,b}), (p_{1,b} - p_{1,a} + p_{2,b} - p_{2,a})\}$. To process job a earlier than job b, both elements of $2 \times (p_{1,a} - p_{1,b})$ and $(p_{1,b} - p_{1,a} + p_{2,b} - p_{2,a})$ should be less than or equal to zero. If $p_{1,a} - p_{1,b} \leq 0$ in the first element, then $p_{1,b} - p_{1,a} \geq 0$, thus, $p_{2,b} - p_{2,a}$ must be less than or equal to zero in the second element, that is $p_{2,b} - p_{2,a} \leq 0$. This last inequality is in conflict to the condition of $p_{2,a} - p_{2,b} \leq 0$ generated from the source of processing times. If machine 2 represents ORs, the condition of $p_{2,a} - p_{2,b} \leq 0$ relates to the SPT rule and $p_{2,b} - p_{2,a} \leq 0$ relates to the LPT rule.

If the source of processing times affects the total completion time more than the source of idle times, then we should follow the condition of $p_{2,a} - p_{2,b} \leq 0$ to min (ΣC_j) for n-job 2-machine flow shop problems, although it creates more idle times on machine 2. However, $C_{max} = \sum_{j=1}^{n} X_{m,j} + \sum_{j=1}^{n} p_{m,j}$, where m is the last machine and $\sum_{j=1}^{n} p_{m,j}$ is a constant, which means the objective of min (C_{max}) focuses only on idle times [10]. If processing times dominate idle times on the performance of total completion time, then min (ΣC_j) and min (C_{max}) are in conflict, which means minimization of one objective will result in maximization of the other. Only when idle times dominate processing times on the performance of total completion time, are min (ΣC_j) and min (C_{max}) consistent with each other.

4. A New Objective for OR Scheduling and A Heuristic

4.1 A New Objective for OR scheduling

Wait list minimization in OR slate scheduling can be achieved by fulfilling either objective used in flow shop production scheduling. One objective is maximum completion time minimization, min (C_{max}), and the other objective is total completion time minimization, min (ΣC_j). For general n-job m-machine flow shop production problems, these two objectives are inconsistent with each other on machines from the second to the last. OR slate scheduling can be modeled as a 3-machine flow shop, where a job represents a surgical case and a machine represents one of three perioperative units – a pre-op unit (bed), an intraoperative unit (operating room), or a post-op unit (PACU bed). Therefore, we propose a new method for scheduling OR slates, taking both C_{max} and ΣC_j into consideration.

$$\min\left(\sum_{i=2}^{m} C_{max,i} + \sum_{i=1}^{m}\sum_{j=1}^{n} C_{i,j}\right) \tag{3}$$

For flow shop production, we assume that all jobs are available at time zero for processing on machine 1, which means there is no idle time on machine 1. Consequently, the maximum completion time on machine 1 is a constant, which is the sum of processing times of all n jobs on machine 1. Therefore, we use Equation (3) to evaluate the performance of a heuristic for OR slate scheduling, instead of using min $\sum_{i=1}^{m}\left(C_{max,i} + \sum_{j=1}^{n} C_{i,j}\right)$ that includes C_{max} on machine 1 into evaluation.

Because of the interdependencies that exist among the perioperative units, Equation (3) evaluates not only the performance of operating room slates, but also the scheduling of corresponding pre-op and post-op beds. Thus we propose a third objective that takes the prior two objectives into consideration simultaneously.

4.2 A State Space with Head, Body, and Tail Heuristic

Idle times affect the maximum completion time on $m - 1$ (peri-op) units and the total completion time on m units (a.k.a. machines in a flow shop). We know that we need to put small cases in the head (front) and tail (back) of a sequence. Therefore, we use the enumeration method to evaluate combinations of head and tail cases (jobs), that is to put each of n cases into the head of a sequence, and each of the remaining $n - 1$ cases into the tail of a sequence, and then to choose the combination of head and tail cases with the best performance on Equation (3). To reduce the computational complexity of our heuristic, we use average processing times (*APT*s) [19] to represent cases in the body of a sequence, which are cases between the head and tail of a sequence. Next, we propose our state space with head, body, and tail (SS-HBT) heuristic in a perioperative context.

For an n-case m-unit instance, our SS-HBT heuristic uses r = *round down* $(n/2)$ rounds to construct a sequence, and for each round, SS-HBT uses the following steps.

Step 1 Enumerate all combinations of head and tail cases for unselected cases. In each round, there will be $n - (r - 1) \times 2$ cases left.

Step 2 Calculate *APT*s for cases in the body of a sequence according to the combination of head and tail cases.

Step 3 Calculate the completion time of each case for each (peri-op) unit for each combination of head, body, and tail cases, in which *APT*s represent the body (intraoperative) cases. This step is to calculate states, i.e., completion times. For details about the state space concept, please refer to [20].

Step 4 Choose candidate sequences with the best performance on Equation (3), which are the combinations of head and tail cases, and send the candidates and relative states to the next round.

In each round, the SS-HBT heuristic selects two cases, one for the head of a sequence and the other for the tail of a sequence. When r rounds are finished, there are zero cases or one case left unselected, depending on whether n is even or odd. When no case is left, the states are correct for a candidate slate in the last round, and we can output the best candidate slates and states. When only one case is left unselected, we can put such case in the only position in a candidate sequence, and output the finished sequence (slate) and the states in the last round. Such states in the last round are correct, because the *APT*s of the body case accurately represent processing times of the only unselected case.

Table 1: A 4-case 3-unit instance

Case	Pre-op	OR	Post-op
1	2	2	6
2	6	8	4
3	7	6	3
4	7	4	4

To illustrate the procedure of our SS-HBT heuristic, we use a 4-case (job) 3-unit (machine) instance as in Table 1. For the initial round, $r = 1$, there are 4 cases unselected, and there are $4 \times 3 = 12$ combinations of head and tail cases. If cases 1 and 4 are selected as head and tail cases respectively, the *APT*s are (6.5, 7, 3.5) for unselected cases 2 and 3 on each unit , that is 6.5 on unit 1 (pre-op bed), 7 on unit 2 (OR), and 3.5 on unit 3 (post-op bed). The performance on Equation (3) for a case sequence of (1, *APT*, *APT*, 4) is 258.5, which is the best outcome in round 1. Actually, in round 1 for such instance, the head-tail combinations of cases 1&3 and cases 1&4 have the same performance. These two candidate sequences and states are sent to the next round. However, in round 2, the sequence of (1, 2, 3, 4) performs better than any other combinations, with an outcome of 256. Finally, the SS-HBT heuristic outputs the sequence of (1, 2, 3, 4), its performance of 256, and relative states.

5. Case Studies

To test the performance of our SS-HBT heuristic, we randomly generate a number of small instances. The case (job) number and unit (machine) number are 5, 6, 7, or 8. Thus, there are 16 scales of instances, varying from 5-case 5-unit instances to 8-case 8-unit instances. Without loss of generality, the processing times are integers, normally distributed in a range from 1 to 99 inclusively, and we generate 20 instances for each scale. Therefore, there are 320 small instances in total.

In this section, we first show the inconsistency and consistency among the three objectives, min (C_{max}), min (ΣC_j) and min ($\Sigma C_{max,i} + \Sigma\Sigma C_{i,j}$), by comparing the performance of a sequence on each of the three objectives. Secondly, we compare the performance of SPT and LPT rules on the three objectives respectively. Then, we compare our heuristic with the NEH and CDS heuristics on the three objectives.

5.1 Inconsistency and Consistency among Three Objectives

Using the exhaustive enumeration method for all small instances, we can get the optimal solutions on each of the three objectives, respectively. An optimal solution refers to the sequence and its performance on an objective. However, there is inconsistency among the three objectives. It means that, for example, the optimal sequence on min (C_{max}) might not have the optimal performance on min (ΣC_j) or min ($\Sigma C_{max,i} + \Sigma\Sigma C_{i,j}$), which is the same for the other two objectives. Therefore, to illustrate how much inconsistency is among the three objectives, we compare the performance of an optimal sequence on one objective with its performance on the other two objectives (Table 2).

The first column in Table 2 indicates the scale of instances. INT means processing times are integers, the first digit means a number of cases, and the second digit means a number of units. For example, INT5×6 means 5 cases and 6 units. The second and third columns are for optimal sequences on min (C_{max}), but indicate the performance deviations of such sequences to min (ΣC_j) and to our objective min ($\Sigma C_{max,i} + \Sigma\Sigma C_{i,j}$) respectively. The fourth and fifth columns are for optimal sequences on min (ΣC_j), but indicate the deviations to min (C_{max}) and to our objective respectively; and the sixth and seventh columns are for optimal sequences on our objective, but indicate the deviations to min (C_{max}) and to min (ΣC_j) respectively. The numbers in relative columns are average deviations in percentage for 20 instances in a scale. Max and Min indicate the maximum and minimum deviations among all of 320 instances. Average means the average deviation for all 320 instances. Sum is the summation of deviations for one objective to the other two objectives.

From the average deviations in Table 2, we can see optimal sequences on min (ΣC_j) can achieve a smaller deviation of 1.54% on average to our objective, and those on our objective achieve a larger deviation of 1.76% on average to min (ΣC_j). However, the deviation of 9.62% on average for min (ΣC_j) to min (C_{max}) is much larger than that of 5.31% for our objective to min (C_{max}). Obviously in this sense, min (ΣC_j) and min (C_{max}) are inconsistent with each other. The average deviation of min (C_{max}) to min (ΣC_j) is 7.76%, and that of min (ΣC_j) to min (C_{max}) is 9.62%. By the sum of deviations, it shows that our objective is compatible with min (ΣC_j) and min (C_{max}), which is 7.06% for our objective versus 13.23% for min (C_{max}), and 11.15% for min (ΣC_j).

Table 2: Inconsistency among min (C_{max}), min (ΣC_j), and our *Obj* (%)

Instance	min (C_{max})		min (ΣC_j)		Our *Obj*	
	min(ΣC_j)	Our *Obj*	min(C_{max})	Our *Obj*	min(C_{max})	min(ΣC_j)
INT5×3	10.55	6.43	10.25	0.96	5.04	1.14
INT5×4	4.57	2.14	8.90	1.00	3.25	1.43
INT5×5	6.83	3.15	10.40	1.58	3.66	1.91
INT5×6	6.94	4.69	8.54	1.96	5.20	1.82
INT5×7	6.34	3.84	8.22	1.38	5.58	1.26
INT5×8	6.04	3.64	7.67	1.28	4.62	2.16
INT6×3	10.22	6.80	8.86	0.93	3.95	1.40
INT6×4	13.51	10.08	9.59	1.32	7.29	1.60
INT6×5	7.08	4.49	8.09	1.46	4.51	1.59
INT6×6	7.30	5.54	7.72	1.27	5.31	1.72
INT6×7	8.31	6.48	7.08	1.26	4.91	2.04
INT6×8	5.27	3.82	9.72	1.50	4.34	1.61
INT7×3	10.07	7.49	9.64	1.23	4.50	1.87
INT7×4	9.60	8.50	9.37	1.69	4.73	1.65
INT7×5	6.30	5.19	10.42	1.98	6.74	1.29
INT7×6	6.73	4.91	10.44	1.61	6.19	1.53
INT7×7	6.83	3.59	10.76	2.03	5.63	2.48
INT7×8	5.68	4.03	8.16	1.28	6.04	1.39
INT8×3	10.91	8.52	9.60	1.81	4.33	1.76
INT8×4	9.91	8.34	9.34	1.43	4.86	2.08
INT8×5	7.86	5.90	12.92	1.71	6.86	1.42
INT8×6	6.22	3.98	13.72	2.18	5.75	2.52
INT8×7	6.73	5.22	10.62	2.04	7.50	2.07
INT8×8	6.49	4.49	10.78	2.00%	6.61%	2.41
Max	39.53	31.34	34.14	10.30	27.10	13.83
Min	0.00	0.00	0.00	0.00	0.00	0.00
Average	7.76	5.47	9.62	1.54	5.31	1.76
Sum	13.23		11.15		7.06	

We also want to point out that the minimum deviation is 0.00% for each of the three objectives to each other, which means two objectives are consistent with each other for some instances, that is, an optimal sequence on one objective can also achieve the optimal performance on another objective. Table 3 shows the consistency among the three objectives.

If an optimal sequence on one objective can also achieve the optimal performance on another objective for an instance, then we count one for the original objective. Counting the consistency this way, we can see in Table 3 that, among 320 instances, optimal sequences on min (C_{max}) also achieve optimal performance on min (ΣC_j) for 11 instances and on our objective for 30 instances, for optimal sequences on min (ΣC_j), optimal on min (C_{max}) for 26 instances and on our objective for 95 instances, and for optimal sequences on our objective, optimal on min (C_{max}) for 60 instances and on min (ΣC_j) for 95 instances. Overall, optimal sequences on our objective achieve the consistency with the other two objectives for 155 instances in total, which is better than the consistency of the other two objectives, 41 instances in total for min (C_{max}) and 121 instances in total for min (ΣC_j).

Consequently, we need to achieve the objective of min ($\Sigma C_{max,i} + \Sigma\Sigma C_{i,j}$) for scheduling OR slates to minimize the wait list, instead of achieving min (C_{max}) or min (ΣC_j) individually. To be more specific, we need to focus on the scheduling of all units in a hospital, instead of scheduling of operating rooms only.

Table 3: Consistency among min (C_{max}), min (ΣC_j), and our *Obj*

Instance	min (C_{max})		min (ΣC_j)		Our *Obj*	
	min(ΣC_j)	Our *Obj*	min(C_{max})	Our *Obj*	min(C_{max})	min(ΣC_j)
INT5×3	1	5	3	8	6	8
INT5×4	1	5	1	7	8	7
INT5×5	0	6	1	4	9	4
INT5×6	1	1	3	5	2	5
INT5×7	2	3	4	6	4	6
INT5×8	2	2	3	6	3	7
INT6×3	0	0	1	8	4	7
INT6×4	0	0	0	9	0	9
INT6×5	1	1	2	4	5	4
INT6×6	2	4	4	5	6	5
INT6×7	1	1	1	7	1	7
INT6×8	1	2	1	6	3	5
INT7×3	0	1	2	3	3	3
INT7×4	0	0	0	5	2	5
INT7×5	0	1	0	5	1	5
INT7×6	0	1	0	3	1	3
INT7×7	1	2	1	4	2	4
INT7×8	0	1	1	4	2	4
INT8×3	0	0	1	2	5	2
INT8×4	0	0	1	2	3	3
INT8×5	0	0	0	3	0	3
INT8×6	0	2	0	1	2	1
INT8×7	0	2	0	0	2	0
INT8×8	0	0	0	3	0	3
Sum	11	30	26	95	60	95
Total	41		121		155	

5.2 The SPT Rule vs. the LPT Rule

The SPT and LPT rules are conflicting in nature in sequencing cases [10], and have conflicting OR outcomes [3-7]. Hospitals would like to achieve high OR utilization and use the LPT rule in scheduling OR slates. From the perspective of flow shop production scheduling, the SPT rule is good to minimize flow time [10, 21]. Both high OR utilization and small flow time, however, are good to minimize the wait list for the OR slate. To clearly show the conflict, we compare the performance of SPT and LPT rules on the three objectives respectively (Table 4).

We can see from Table 4 that a deviation of 18.62% on average for the LPT rule on min (C_{max}) is smaller than that of 19.15% for the SPT rule. This confirms that, on average, the LPT rule can achieve higher OR utilization than the SPT rule. However, on average, the SPT rule achieves a deviation of 6.82% on min (ΣC_j) much smaller than that of 32.94% for the LPT rule, which confirms that the SPT rule is good to minimize flow time. Moreover, the SPT rule achieves a deviation of 8.96% on our objective, which is smaller than that of 30.09% for the LPT rule.

From the perspective of flow shop production scheduling, the performance of the SPT and LPT rules on min (C_{max}) are not acceptable. According to Ruiz and Maroto [17], the SPT and LPT rules are the worst among 19 heuristics to minimize the maximum completion time, with average deviations more than 24% on Taillard's benchmarks. Therefore, we need to seek other heuristics for OR scheduling.

Table 4: Performance of SPT & LPT rules on min (C_{max}), min (ΣC_j), & our *Obj* (%)

Instance	SPT Rule			LPT Rule		
	min(C_{max})	min(ΣC_j)	Our *Obj*	min(C_{max})	min(ΣC_j)	Our *Obj*
INT5×3	19.90	5.38	7.26	17.22	34.82	31.27
INT5×4	16.63	4.40	6.00	19.74	30.33	28.85
INT5×5	17.15	4.57	7.25	11.88	27.64	22.39
INT5×6	17.05	4.57	8.36	16.33	27.31	25.73
INT5×7	19.27	5.90	9.54	15.61	25.85	23.92
INT5×8	15.50	4.14	6.48	12.93	24.92	23.70
INT6×3	14.24	5.99	6.74	21.32	46.98	39.74
INT6×4	14.93	4.60	6.07	20.57	46.45	37.43
INT6×5	17.13	5.48	8.22	17.01	27.71	26.76
INT6×6	15.82	5.29	7.65	15.19	27.76	25.05
INT6×7	14.86	4.99	7.48	22.00	34.29	31.94
INT6×8	19.13	6.63	9.50	17.19	26.70	25.91
INT7×3	19.57	9.39	10.27	17.50	39.61	36.29
INT7×4	19.86	8.07	8.89	21.37	39.06	36.16
INT7×5	22.48	7.70	10.23	19.28	29.54	26.80
INT7×6	17.95	5.79	7.79	22.01	33.16	31.42
INT7×7	21.35	6.74	9.91	19.07	29.64	27.07
INT7×8	21.46	7.53	11.10	20.95	28.27	28.00
INT8×3	19.74	9.41	9.55	16.12	43.11	38.24
INT8×4	19.99	10.80	9.61	20.76	39.79	35.04
INT8×5	21.81	8.67	10.22	18.64	33.50	29.80
INT8×6	28.43	10.25	13.72	21.80	30.28	28.29
INT8×7	23.34	7.79	10.25	19.75	31.05	29.53
INT8×8	22.11	9.69	12.90	22.52	32.72	32.85
Max	51.77	31.06	28.58	47.87	106.57	87.57
Min	0.00	0.00	0.00	0.00	0.00	0.69
Average	19.15	6.82	8.96	18.62	32.94	30.09

5.3 The NEH, CDS, and SS-HBT Heuristics

According to Ruiz and Maroto [17], the NEH heuristic is the best among 19 heuristics for flow shop production scheduling to min (C_{max}), the CDS heuristic is the eighth, and the SPT and LPT rules are the worst. The NEH heuristic takes the LPT rule into consideration, and the CDS heuristic takes the SPT rule into consideration. We compare the performance of our SS-HBT heuristic with the NEH and CDS heuristics on the three objectives respectively (Table 5).

In Table 5, Max and Min represent the maximum and minimum deviations among all of 320 instances, respectively. We can see that the NEH heuristic achieves the smallest deviation of 1.12% on average to min (C_{max}), where it is 3.82% for the CDS heuristic and 6.66% for our SS-HBT heuristic. However, our SS-HBT heuristic achieves the smallest deviation of 3.05% on average to min (ΣC_j), where it is 7.30% for the NEH heuristic and 6.30% for the CDS heuristic. The deviation of 3.05% for our SS-HBT heuristic to min (ΣC_j) is smaller than that of 6.82% for the SPT rule. Our heuristic is good to minimize flow time. Moreover, taking input and output facilities into consideration for OR scheduling, our objective is necessary to minimize the wait list in OR scheduling. The SS-HBT heuristic achieves the smallest deviation of 1.00% on average to our objective, where it is 4.98% and 3.42% for the NEH and CDS heuristics respectively.

Table 5: NEH, CDS, and SS-HBT on min (C_{max}), min (ΣC_j), and our *Obj* (%)

Instance	NEH			CDS			SS-HBT		
	min(C_{max})	min(ΣC_j)	Our *Obj*	min(C_{max})	min(ΣC_j)	Our *Obj*	min(C_{max})	min(ΣC_j)	Our *Obj*
INT5×3	0.23	8.26	4.97	1.55	6.53	2.93	6.30	1.94	0.27
INT5×4	0.81	4.11	2.59	2.97	5.43	2.76	4.69	1.26	0.26
INT5×5	0.98	5.55	3.21	2.25	3.63	1.49	4.29	3.48	0.75
INT5×6	0.77	6.76	4.10	3.25	3.29	1.79	5.29	2.48	0.37
INT5×7	0.92	6.41	4.27	3.01	3.73	2.07	5.32	2.30	0.55
INT5×8	0.80	5.95	4.03	2.13	2.47	1.71	5.22	2.20	0.50
INT6×3	0.17	8.67	5.66	3.45	10.53	4.76	4.48	1.84	0.42
INT6×4	0.69	10.28	6.77	2.64	7.16	3.26	9.34	3.66	1.54
INT6×5	0.62	6.72	3.73	2.56	4.85	2.48	5.63	2.32	0.71
INT6×6	0.99	7.18	5.28	3.15	4.33	2.99	6.83	2.55	0.96
INT6×7	1.44	8.53	5.96	3.31	2.80	1.48	6.77	3.69	1.08
INT6×8	0.79	4.70	3.87	2.69	3.62	2.71	6.52	3.17	1.47
INT7×3	1.00	8.93	5.69	3.59	12.71	5.73	4.84	2.59	0.29
INT7×4	0.82	7.20	5.44	4.18	6.71	4.56	6.10	2.87	0.88
INT7×5	1.54	7.44	5.33	6.63	7.12	4.24	7.82	2.89	0.82
INT7×6	1.87	6.70	5.69	5.70	5.62	3.34	6.15	3.17	1.25
INT7×7	1.16	7.19	4.45	4.46	5.79	3.18	6.23	4.40	1.50
INT7×8	1.57	6.86	5.55	3.81	4.40	3.20	8.11	2.94	1.29
INT8×3	0.31	9.39	5.51	2.06	12.96	6.59	6.36	2.99	0.76
INT8×4	1.25	7.94	5.80	4.53	9.46	4.67	7.44	3.32	1.04
INT8×5	2.16	8.60	5.82	4.86	8.02	4.33	6.73	3.03	1.26
INT8×6	1.94	6.95	4.58	5.38	6.28	3.65	9.82	5.06	2.14
INT8×7	2.45	7.42	5.85	7.77	6.59	4.09	9.51	4.03	1.94
INT8×8	1.62	7.39	5.38	5.73	7.11	3.98	9.95	4.93	1.97
Max	9.13	31.46	24.89	16.45	41.13	26.40	27.78	17.29	6.93
Min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average	1.12	7.30	4.98	3.82	6.30	3.42	6.66	3.05	1.00

6 Conclusion and Future Work

Wait list minimization is a broadly defined goal for OR slate scheduling. It can be achieved from many perspectives, for example by increasing resources, i.e., more surgeons, more beds, longer daily block times of operating rooms, etc. If the original utilization of resources is low, then adding more resources would make it lower. Moreover, the scheduling of pre-op units, operating rooms, and post-op units should be in a coordinated manner. Otherwise, smooth patient and resource flows are compromised. This is why some scholars recommend the SPT rule in OR scheduling to minimize flow time, although the LPT rule can achieve high OR utilization.

In summary, two specific objectives in flow shop production scheduling are applicable to wait list minimization in the context of scheduling OR slates. These objectives are to minimize maximum completion time, min (C_{max}), and to minimize total completion time, min (ΣC_j). Minimizations of maximum completion time and total completion time are related to improvements of OR utilization and patient flow in a hospital respectively. We show that these two objectives are inconsistent for general cases, although total completion time includes maximum completion time. Therefore we propose a new method for scheduling OR slates that minimize the wait list, min ($\Sigma C_{max,i} + \Sigma\Sigma C_{i,j}$), while focusing on the scheduling of the primary perioperative services (the pre-op unit, operating rooms, and post-op or

PACU). Moreover, extending research on flow shop production scheduling, we also provide a SS-HBT heuristic that outperforms the NEH and CDS heuristics for determining OR slates.

There are many factors in a hospital affecting the performance of operating rooms, such as variation in surgery times, block times of operating rooms, surgeon schedules, the functionalities of operating rooms vs. surgery types, etc. Moreover, OR planning in the long term will affect OR scheduling in short term and adaptive OR control in real time. How the scheduling of operating rooms in the long term affects scheduling in short term is the subject of our future research.

Acknowledgements

This project was supported by MITACS-Accelerate and Foothills Medical Centre, Calgary, Alberta, Canada

References

1. Testi, A., Tanfani, E., and Torre, G., 2007, "A three-phase approach for operating theatre schedules," *Health Care Management Science*, 10 (2), 163-172.
2. Vanberkel, P.T., Boucherie, R.J., Hans, E.W., Hurink, J.L., van Lent, W.A.M., and van Harten, W.H., 2011, "Accounting for inpatient wards when developing mater surgical schedules," *Anesthesia and Analgesia*, 112 (6), 1472-1479.
3. Magerlein, J.M., and Martin, J.B., 1978, "Surgical demand scheduling: A review," *Health Services Research*, 13 (4), 418-433.
4. Cardoen, B., Demeulemmeester, E., and Belien, J., 2010, "Operating room planning and scheduling: A literature review," *European Journal of Operational Research*, 201 (3), 921-932.
5. Gupta, D., and Denton, B., 2008, "Appointment scheduling in health care: Challenges and opportunities," *IIE Transactions*, 40 (9), 800-819.
6. Goldman, J., Knappenberger, H.A., and Moore, E.W., 1969, "An evaluation of operating room scheduling policies," *Hospital Management*, 107 (4), 40-51.
7. Kwak, N.K., Kuzdrall, P.J., and Schmitz, H.H., 1976, "The GPSS simulation of scheduling policies for surgical patients," *Management Science*, 22 (9), 982-989.
8. Marcon, E., and Dexter, F., 2006, "Impact of Surgical Sequencing on Post Anesthesia Care Unit Staffing," *Health Care Management Science*, 9 (1), 87-98.
9. Gul, S., Denton, B.T., Fowler, J.W., and Huschka, T., 2011, "Bi-criteria Scheduling of Surgical Services for an Outpatient Procedure Center," *Production and Operations Management*, 20 (3), 406-417.
10. Pinedo, M., 2002, *Scheduling Theory, Algorithms, and Systems*, Englewood Cliffs, Prentice Hall, New Jersey.
11. Nawaz, M., Enscore, E.E., and Ham, I., 1980, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", *Omega-International Journal of Management Science*, 11 (1), 91-95.
12. Campbell H.G., Dudek, R.A., and Smith, M.L., 1970, "A heuristic algorithm for the n-job, m-machine scheduling problem," *Management Science*, 16 (10), 630-637.
13. Gupta, J.N.D., and Staford Jr., E.F., 2006, "Flowshop research after five decades," *European Journal of Operational Research*, 69 (3), 699-711.
14. Garey, M.R., Johnson, D.S., and Sethi, R., 1976, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, 1 (2), 117-121.
15. Garey, M.R., and Johnson, D.S., 1979, *Computers and intractability*, Freeman, San Francisco, California.
16. Hoogenveen, J.A., and Kawaguchi, T., 1999, "Minimizing total completion time in a two-machine flowshop: analysis of special cases," *Mathematics of Operations Research*, 24 (4), 887-910.
17. Ruiz, R., and Maroto, C., 2005, "A comprehensive review and evaluation of permutation flowshop heuristics," *European Journal of Operational Research*, 165 (2), 479-494.
18. Botta-Genoulaz, V., 2000, "Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness," *International Journal of Production Economics*, 64 (1), 101–111.
19. Li, W., Luo, X.G., Xue, D., and Tu, Y.L., 2011, "A heuristic for adaptive production scheduling and control in flow shop production," *International Journal of Production Research*, 49 (11) 3151-3170.
20. Li, W., Nault, B.R., Xue, D., and Tu, Y.L., 2011, "An efficient heuristic for adaptive production scheduling and control in one-of-a-kind production," *Computers & Operations Research*, 38 (1), 267-276.
21. Conway, R.W., Maxwell, W.L., and Miller, L.W., 1967, *Theory of Scheduling*, Addison-Wesley, Massachusetts.