IIE Transactions

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uiie20

The extraboard operator scheduling and work assignment problem

Fei Li\textsuperscript{a} & Diwakar Gupta\textsuperscript{a}

\textsuperscript{a} Industrial & Systems Engineering Department, University of Minnesota, Minneapolis, MN 55455, USA E-mail:
Accepted author version posted online: 27 Jan 2014. Published online: 28 Jul 2014.

To cite this article: Fei Li & Diwakar Gupta (2014) The extraboard operator scheduling and work assignment problem, IIE Transactions, 46:11, 1132-1146, DOI: 10.1080/0740817X.2014.882036

To link to this article: http://dx.doi.org/10.1080/0740817X.2014.882036

PLEASE SCROLL DOWN FOR ARTICLE

Taylor & Francis makes every effort to ensure the accuracy of all the information (the “Content”) contained in the publications on our platform. However, Taylor & Francis, our agents, and our licensors make no representations or warranties whatsoever as to the accuracy, completeness, or suitability for any purpose of the Content. Any opinions and views expressed in this publication are the opinions and views of the authors, and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content should not be relied upon and should be independently verified with primary sources of information. Taylor and Francis shall not be liable for any losses, actions, claims, proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or howsoever caused arising directly or indirectly in connection with, in relation to or arising out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms & Conditions of access and use can be found at http://www.tandfonline.com/page/terms-and-conditions
The extraboard operator scheduling and work assignment problem

FEI LI AND DIWAKAR GUPTA*

Industrial & Systems Engineering Department, University of Minnesota, Minneapolis, MN 55455, USA
E-mail: guptad@me.umn.edu

Received November 2012 and accepted November 2013

An instance of the operational fixed job scheduling problem arises when open work caused by unplanned events such as bus breakdowns, inclement weather, and driver (operator) absenteeism need to be covered by reserve (extraboard) drivers. Each work-piece, which is referred to as a job, requires one operator who must work continuously between specified start and end times to complete the job. Each extraboard operator may be assigned up to $w$ hours of work, which may not to be continuous so long as the total work time is within a $s$-hour time window of that operator’s shift start time. Parameters $w$ and $s$ are called allowable work-time and spread-time, respectively. The objective is to choose operators’ shift start times and work assignments, while honoring work-time and spread-time constraints, such that the amount of work covered as part of regular duties is maximized. This paper argues that the extraboard operator scheduling problem is NP-hard and three heuristic approaches are presented for the solution of such problems. These include a decomposition-based algorithm whose worst-case performance ratio is proved to lie in $[1 - 1/e, 19/27]$, where $e \approx 2.718$ is the base of the natural logarithm. Numerical experiments are presented that use data from a large transit agency, which show that the average performance of the decomposition algorithm is good when applied to real-world data.

**Keywords:** Operational fixed job scheduling problem, work-time and spread-time constraints, decomposition algorithm, approximation ratio

1. Introduction

The Fixed Job Scheduling (FJS) problem, introduced in Gertsbakh and Stern (1978), concerns the optimal assignment of jobs to operators, where each job has a fixed start time and a fixed end time, and each operator can process at most one job at a time (Kolen et al., 2007). Instances of the FJS problem arise in many applications. For example, the problem of scheduling aircraft maintenance jobs that are required to be completed within fixed time windows (Kroon et al., 1995) and the problem of scheduling bus operators (Martello and Toth, 1986) are both instances of the FJS problem.

There are two broad categories of the FJS problem: tactical and operational. In the Tactical FJS (denoted as TFJS) problem the objective is to determine the minimum number of operators needed to cover all jobs. There is no a priori limit on the number of available operators. In contrast, in the Operational FJS (denoted as OFJS) problem, the number of available operators is fixed and the objective is to maximize a total reward. In this setting, the assignment of a job to an available operator produces a reward (usually proportional to its duration), whereas jobs that remain unassigned do not produce a reward.

Spread-time and work-time constraints are two major types of constraints in FJS problems. Spread-time is the maximum time span of an operator's workday. That is, given a spread-time limit $s$, if an operator is scheduled to start work at $t_0$ on a particular day, then she or he may be assigned work between $t_0$ and $(t_0 + s)$ but not outside this interval of time. Work-time is the maximum amount of time that an operator may be required to work within the allowed spread-time. We denote work-time limit by $w$. Operators may voluntarily choose to work more than $w$ hours each day, but in such cases they receive overtime pay. The hourly overtime pay is higher than the hourly regular pay. We affix letters $S$ and $W$ to TFJS or OFJS to identify problem instances with appropriate constraints. For example, OFJS-S represents the OFJS problem with spread-time constraints only (i.e., $w = s$), TJFS-W means the TFJS problem with work-time constraints only (i.e., $w$ is finite but $s$ can be arbitrarily large), and OFJS-WS means the OFJS problem with both types of constraints (i.e., $w \leq s < \infty$). Note that $w$ cannot exceed $s$ if a spread-time limit is specified.

This paper is motivated by an instance of the OFJS-WS problem that arises in the context of extraboard bus...
Operator scheduling and work assignment problem

There are not assigned to extraboard operators are performed by bus drivers in overtime. Because ample availability of overtime was observed in data from a collaborating transit agency, we do not model cases in which service may be dropped.

A different way to understand the scope of the problem we study in this paper is to place it within the hierarchy of extraboard workforce planning and management problems consisting of operational, tactical, and strategic levels; see, for example, Koutsopoulos (1990). Within this hierarchy, we focus on dispatch decisions that belong to the lowest—i.e., the operational—level. Examples from the other two levels include extraboard workforce sizing, run cutting methods, and the determination of the daily number of operators who would be scheduled to serve as extraboard. MacDorman and MacDorman (1982), Perry and Long (1984), and MacDorman (1985) contain additional institutional background on workforce management challenges in the context of transit operations.

Instances of the OFJS-WS problem arise in many application areas and are of general interest to operations engineers and managers. First, we draw attention to the fact that all types of public transportation operations need extra operators to take care of contingencies and avoid gaps in service. Examples include bus, rail, ferry, and passenger airline operations. Work assignment problems similar to what we study in this paper arise in each of these settings. In addition, OFJS-WS problems arise in the context of periodic batch scheduling of jobs on parallel machines. Because jobs and machines can represent different entities in different application areas, there are numerous applications of the OFJS-WS problem. For example, jobs could be groups of orders that need machining or repair, deferrable surgeries that need operating room time, or computer programs that need processor time.

Eliyi and Azizoglu (2006) show that the OFJS-S problem is NP-hard by arguing first that one can solve any instance of the TFJS problem by repeatedly solving the same instance of the OFJS problem with 1, . . . , m operators, where m is the number of jobs. In the previous sentence, the words “same instance” mean an instance of the problem with the same set of jobs, and w and s, if w and s are specified. The NP-hardness of the OFJS-S problem then follows from Fischetti et al. (1987). Similar arguments exist for OFJS-W and OFJS-WS problem instances as well. As the OFJS-W, OFJS-S, and OFJS-WS problems are all NP-hard, it is difficult to compare the relative difficulty of solving each problem. Therefore, to explain the need for focusing attention on the OFJS-WS problem, we present two arguments. First, the OFJS-WS problem is different from the OFJS-W problem because it also considers spread constraints. It is also different from the OFJS-S problem because the work-time constraints limit the amount of work that may be assigned within a spread. Therefore, solution methods developed in previous studies, discussed below, are not applicable to OFJS-WS problems. Second, OFJS-S and OFJS-W problems are both special cases of the OFJS-WS
problem and only the latter captures the actual constraints faced by transit agencies.

Fischetti et al. (2007) provide a review of the fixed job scheduling literature, which is also referred to as the interval scheduling problem. The authors divide papers into four groups based on model features and objective. These categories are as follows.

1. All jobs must be performed and the objective is to minimize the number of machines used.
2. The number of machines is fixed and the objective is to maximize total weight of jobs assigned.
3. Job start times are not fixed, and the objective is either to maximize total weight or number of jobs or to minimize the number of machines used to cover all jobs.
4. Jobs are scheduled online (one at a time) or previously scheduled jobs may be preempted, and the objective is either one of the two objectives mentioned in category 3.

Our study falls into the second group above. In what follows, we discuss key papers belonging to groups 1, 2, 4.

We do not discuss papers belonging to group 3 because fixed start and end times of jobs is an important feature of OSJF-WS problems and methods that do not assume fixed start/end times are not relevant in our setting.

Significant contributions in the first group include Fischetti et al. (1987, 1989, 1992) and Martello and Toth (1986). Fischetti et al. (1987) show that the TFJS-S problem is NP-hard. Fischetti et al. (1989) study the TFJS-W, show that it is NP-hard, prove a 2-upper-bounding property of its preemptive version, and provide a branch-and-bound algorithm to solve it. Fischetti et al. (1992) propose several approximation algorithms for solving different versions of the TFJS problem, including greedy algorithms and preemption-based algorithms. Martello and Toth (1986) study the bus driver scheduling problem, which can be considered as an instance of the TFJS problem with work- and spread-time constraints, relief point constraints, as well as other work rules but do not provide an algorithm with a guaranteed approximation ratio. As the tactical version of the FJS problem is different from the operational version, the above algorithms do not apply to the extraboard driver scheduling problem we consider. In addition, there are a variety of applied papers on the topic. For example, Lourenco et al. (2001) list different objective functions that transit agencies try to optimize and summarize heuristics that have been applied to these problems, including the greedy randomized adaptive search procedure (Feo and Resende, 1995) and genetic algorithms (Sivrikaya-Serifoğlu and Ulusoy, 1999).

However, none of these algorithms provides an approximation ratio for OFJS-WS problem instances, which is the focus of this paper.

Next, we consider the papers in the second group. Arkin and Silverberg (1987) consider scheduling n jobs with fixed start and end times to k non-identical machines with the goal of maximizing the value of all jobs assigned (value could be duration). When machines are identical—i.e., each job can be processed by any machine—the authors argue that the problem can be solved in $O(n^2 \log n)$ time. When machines are not identical—i.e., each job can be processed by a subset of machines (note, processing ability may be the result of available time of each machine)—the authors provide an exact algorithm that runs in $O(n^k)$ time. The main difference is that this formulation assumes that machine availability (i.e., shift start and end times in our setting) is known and that machines do not have both work- and spread-time constraints. We infer the latter implicit assumption from the fact that the authors assume that the subset of machines that can process each job is known. This is only possible when $w = s$ in our setting. Spread- and work-time constraints are two features of our model that are simultaneously important in our setting, which makes our problem formulation different from that in Arkin and Silverberg (1987).

The second group of papers is also related to the k-track assignment problem, in which k machines (possibly with different spreads) are given and the objective is to schedule the maximum number of jobs with fixed start and end times. Brucker and Nordmann (1994) give an $O(n^{k-1}k!(k+1))$ algorithm to solve the standard k-track problem with n jobs and k identical machines. Faigle and Nawijn (1995) provide an optimal online algorithm for the k-track assignment problem with identical time windows. Faigle et al. (1999) provide an online greedy algorithm that guarantees to lose no more than $(k - 1)$ jobs relative to the optimal schedule. However, these algorithms do not apply to our setting because the k-track assignment problem maximizes the total number of jobs assigned, not the total weight or duration of assigned jobs. The solution to the k-track assignment problem would be useful in our setting if all jobs had the same weight. That is not the case. Also, we need to consider both spread- and shift-time constraints of operators, which makes our problem setting different.

Many researchers have proposed algorithms for solving variants of the FJS problem. It is therefore appropriate to ask whether these methods can be adapted to solve the OFJS-WS problem. We argue next that straightforward adaptation will not work for the OFJS-WS problem. Consider, for example, the greedy heuristic included in Fischetti et al. (1992), which is shown to have an approximation ratio of three. Although a greedy approach is a reasonable approach for solving the TFJS problem, it can be arbitrarily bad compared with the optimal solution if applied to certain instances of the OFJS problem. We present arguments to support our claim at a later point in this paper. Similarly, if we were to adapt the branch-and-bound algorithm from Eliyi and Azizoglu (2006), we would encounter a combinatorial number of starting nodes, which would limit the suitability of such approaches when the size of the extraboard workforce is large.

As yet another example, consider the branch-and-price approach in Solyali and Ozpeynirci (2009), which they use
to solve the OFJS-S problem. This algorithm requires the ability to repeatedly solve the one-operator instance of the OFJS-S problem. Unfortunately, this approach is not suitable for the OFJS-WS problem because as we show later in this paper, the OFJS-WS problem with one operator is NP-hard. We also introduce the notion of limited shift splits under which the one-operator case can be solved in polynomial time. Even with the limited shift split requirement in place, the approach would be computationally demanding, requiring $O(m^2)$ operations rather than the $O(m^3)$ operations needed to solve the OFJS-S version of the problem (Solyali and Ozpeynirci, 2009).

More recently, some researchers have focused on online scheduling problems belonging to the fourth group. Bhatia et al. (2007) consider both OFJS-S and TFJS-S problems. For OFJS-S problems, the authors provide a randomized algorithm with expected reward at least $\left(1 - 1/e\right)$ of the optimum, but this performance is not guaranteed in every run of the algorithm. Note that $e \approx 2.718$ is the base of the natural logarithm. Our decomposition algorithm can be applied to OFJS-S problems and it is a deterministic algorithm. That is, its performance does not vary for the same problem parameters and it guarantees a performance of at least $\left(1 - 1/e\right)$ of the optimum every time it is applied. We believe ours is a more implementable and stronger result for the transit agencies’ problem setting.

In this paper we first show that the OFJS-WS problem is NP-hard. Then we provide three heuristics for solving the OFJS-WS problem. The first heuristic is a duration-first greedy algorithm, which assigns the longest unassigned job at each step. We show that the greedy algorithm’s approximation ratio is zero. We next show that the preemptive and partial credit version of the OFJS-S problem is solvable in polynomial time. Combining this result and an algorithm provided by Kroon et al. (1995), we construct a two-stage algorithm. The third algorithm solves the OFJS-WS problem with limited shift splits. We first establish that the one-operator case of the OFJS-WS problem with limited shift splits is polynomially solvable and then prove that a decomposition approach based on maximizing one operator’s assignment at a time has an approximation ratio that lies in $[1 − 1/e, 19/27]$. This algorithm and the approximation ratio also apply to the OFJS-S problem. Thus, our third algorithm improves upon a result reported in Fischetti et al. (1987) about the approximation ratio of an algorithm designed to solve the OFJS-S problem.

The contribution of this paper is threefold: (i) it presents three algorithms for solving the OFJS-WS problem; (ii) it establishes approximation ratios for the recommended decomposition based algorithm; and (iii) it uses real data from a large transit agency to compare the three algorithms. Its methodological novelty lies in developing heuristic methods, analyzing the limited shift splits version of the OFJS-WS problem that is observed in practice, and establishing a deterministic $(1 − 1/e)$-approximation algorithm for that case.

The organization of the remainder of this paper is as follows. In Section 2 we introduce a mathematical formulation of the OFJS-WS problem and establish complexity results. In Section 3 we introduce three heuristics for solving the OFJS-WS problem and investigate whether approximation ratios can be provided in each case. We present numerical experiments that utilize data from a large transit agency in Section 4 and conclude the paper in Section 5. The Online Supplement contains empirical distributions of problem parameters used in numerical experiments.

2. Model formulation, complexity and special cases

Although OFJS-WS is a broad class of problems with many variants, our model formulation and solution methods are motivated by the application domain of extraboard drivers scheduling. For example, we assume that $w$ and $s$ take reasonable finite values and that operators do not take too many (unpaid) splits during a day. That is, our algorithm is allowed to split the working time into only a limited number of pieces within the spread. Additional modeling assumptions are included in this section. For the version of the OFJS-WS problem of practical interest, we provide a $(1 − 1/e)$-approximation algorithm that runs in polynomial time.

In this section we present a mathematical formulation of an instance of the OFJS-WS problem, establish its complexity, and identify special cases that can be solved in polynomial time. We assume that jobs are sorted by start time. The notation used in describing a formal model is presented in Table 1, and key assumptions of the model are as follows.

Table 1. Notation used in formulation

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(0, t_{max})$</td>
<td>start and end time of a day of operation. At the collaborating transit agency, daily operations start at 3:30 am and end at 2:00 am the following day</td>
</tr>
<tr>
<td>$t \in [0, \ldots, t_{max}]$</td>
<td>time index</td>
</tr>
<tr>
<td>$n$</td>
<td>number of operators</td>
</tr>
<tr>
<td>$i \in [1, \ldots, n]$</td>
<td>operator index</td>
</tr>
<tr>
<td>$m$</td>
<td>number of jobs</td>
</tr>
<tr>
<td>$j \in J = [1, \ldots, m]$</td>
<td>job index; $J =$ job index set</td>
</tr>
<tr>
<td>$(s_j, e_j)$</td>
<td>start and end times of job $j$, with $s_1 \leq s_2 \leq \cdots \leq s_m$</td>
</tr>
<tr>
<td>$w$</td>
<td>work-time limit</td>
</tr>
<tr>
<td>$s$</td>
<td>spread-time limit</td>
</tr>
<tr>
<td>$d_j = e_j - s_j$</td>
<td>duration of job $j$</td>
</tr>
<tr>
<td>$I_j(s)$</td>
<td>set of jobs that cannot be assigned to the same operator who performs job $j$</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>binary decision variables; $x_{ij} = 1$ if job $j$ is assigned to operator $i$, and 0 otherwise</td>
</tr>
</tbody>
</table>
Assumption 1. Time is discrete.

Assumption 2. Operators are identical in skill. That is, any operator can perform any job.

Assumption 3. All operators are subject to the same work-time and spread-time limits, denoted by \( w \) and \( s \), respectively.

Assumption 4. Parameter values belong to the following ranges: \( 1 \leq s \leq t_{\text{max}} \) and \( d_j \leq w \leq s \), for all \( j \in J \). This means that no job takes longer than the regular shift length of an operator.

Assumption 5. A job that is not covered by available extra-board operators during their regular work time is assigned on an overtime basis. The cost of overtime is proportional to the duration of the job assigned in overtime.

The above assumptions were supported by data from the collaborating transit agency. For example, 1 minute was the smallest unit of time and jobs whose lengths exceeded the work-time limit of 8 hours were assigned first to those operators who were willing to accept overtime. Assignment of such jobs thus occurred independently of the extra-board operator scheduling and work assignment problem addressed in this paper.

In Table 1, \( I_j(s) \) is the set of jobs that are incompatible with job \( j \), for each \( j \in J \). It contains indices of all jobs that would either overlap with job \( j \) or violate spread-time constraints if offered to the same operator. We are now ready to present a formulation of the OFJS-WS problem:

\[
z = \max_{x_{ij}} \sum_{1 \leq j \leq m} d_j \sum_{1 \leq i \leq n} x_{ij},
\]

subject to:

1. \( \sum_{1 \leq i \leq n} x_{ij} \leq 1, \quad j = 1, \ldots, m \),
2. \( x_{ij} + x_{ik} \leq 1, \quad j = 1, \ldots, m - 1, \quad i = 1, \ldots, n, \quad k \in I_j(s) \),
3. \( \sum_{1 \leq j \leq m} d_j x_{ij} \leq w, \quad i = 1, \ldots, n \),
4. \( x_{ij} \in \{0, 1\} \), \( i = 1, \ldots, n, \quad j = 1, \ldots, m \).

The objective function (1) maximizes total duration of assigned jobs. Recall that the OFJS problem includes dispatch decisions that arise after the number of operators is determined. That is, the wages of extra-board operators are sunk. Therefore, it makes sense to maximize the total amount of work assigned to extra-board operators, which is equivalent to minimizing overtime.

Constraints (2) ensure that each job is assigned no more than once. Constraints (3) guarantee that jobs assigned to the same operator neither overlap nor violate spread-time constraints. Note that we only need to consider \( (m - 1) \) jobs with their incompatible sets because the last job’s incompatible relations are already included in the incompatible sets of jobs whose labels are smaller than \( m \). Constraints (4) are the work-time constraints, and Constraints (5) specify that \( x_{ij} \) variables are binary. In situations where \( w = s \)—i.e., when the problem is an instance of the OFJS-S problem—the above formulation remains intact except that Constraints (4) are no longer needed.

Theorem 1 in Eliiyi and Azizoglu (2006) shows that the OFJS-S problem is NP-hard. The proof of this argument is based on the observation that the OFJS-S problem is NP-hard if the TFJS problem is NP-hard for problems in which \( d_j \leq w, \quad j = 1, \ldots, m \). The condition \( d_j \leq w, \quad j = 1, \ldots, m \) means that we need at most \( m \) operators to cover all \( m \) jobs. From this observation and the fact that the TFJS-S problem is NP-hard (see proof in Fischetti et al., 1987), the authors argue that the OFJS-S problem is NP-hard. By using a similar argument, it follows that the OFJS-W problem is also NP-hard because the TFJS-W problem has been proved to be NP-hard in Fischetti et al. (1989). Finally, the OFJS-WS problem is NP-hard because it includes all instances of the OFJS-W problem as special cases.

2.1. One-operator cases

We have observed that dispatchers rarely assign work in a manner that results in more than one split (scheduled idle period) within an extra-board operator’s work shift. That is, within a spread, operators are usually idled at most once. This is because multiple splits are undesirable from the operators’ viewpoint. Even in situations where more than one split occurs, the maximum number of such splits is bounded and small. That is, limiting splits is a reasonable assumption in the application domain for our model. Therefore, we also analyze problem instances with one operator and a limit on the number of shift splits and show that such problems can be solved in polynomial time. We use the fact that a one-operator \( k \)-split OFJS-WS problem is polynomially solvable to develop a heuristic that decomposes the \( n \)-operator scheduling problem into \( n \) one-operator problems. This heuristic is presented in the Section 3.3. However, we begin this section with the OFJS-WS (unlimited splits) one-operator instance and argue that this version of the problem is NP-hard.

Lemma 1. The one-operator case of the OFJS-WS problem is NP-hard.

Proof. Consider an instance of the subset sum problem with \( m \) items. Let \( d_j \) be item values. The subset problem is to find a subset of the \( m \) items such that sum of their values is equal to \( w \).

Next, consider the following recognition version of the one-operator case of the OFJS-WS problem. For each item \( j \), with duration \( d_j \), start times \( s_j \) are such that \( s_1 = 0, \quad e_j = s_j + d_j \), and \( s_{j+1} = e_j \), for \( j = 1, \ldots, m - 1 \). Let the work-time limit equal \( w \) and spread-time limit equal \( s = e_m - s_1 \). The recognition version of the problem is to find an assignment to one operator such that the
Operator scheduling and work assignment problem

operator's working time is $w$. Note that if the maximization version of the above problem is polynomially solvable, then so is the recognition version. In particular, by solving the maximization version and checking whether the optimal value equals $w$, we solve the recognition version.

In the above instance of the OFJS-WS problem, there are no overlapping jobs and the spread-time limit is large enough that it will not be violated. This means that Constraints (3) and (4) that deal with overlapping jobs and spread-time limit may be removed from the formulation presented in (1) to (5) without affecting this instance of the OFJS-WS problem. At this point, it should be clear that the recognition version of the one-operator case and the subset sum problem are equivalent. Because the subset sum problem is NP-hard (Garey and Johnson, 1979), the one-operator case of the OFJS-WS problem is also NP-hard.

In contrast with Lemma 1, the one-operator case of the OFJS-S problem is polynomially solvable (Eliyi and Azizoglu, 2006). In what follows, we discuss a transformation of the OFJS-S problem to the shortest-path problem, which is utilized in subsequent analysis.

We construct a directed graph utilizing the following steps; see Fig. 1 for an illustration. First, we draw a time line and place $2 \times m$ points on the line. These points represent the start and end times of jobs. The length of the line segment between any two points is the difference between the corresponding time epochs. We also connect each job's start and end times by an additional arc. The weight of this arc is set equal to zero. Note that the direction of each arc is from left to right (start time to end time of each job).

Because there are $m$ jobs, there are at most $m$ choices of shift start times (which are simply the job start times). For each possible shift start time, we cut the graph to match with the shift length and find a shortest path from its leftmost node to its rightmost node. From the shortest path solution, we observe which zero-weight arcs are picked. Those correspond to the jobs that are assigned to the operator. We repeat this procedure for all possible shift start times and the solution with the highest value is the optimal solution of the single-operator OFJS-S problem.

The complexity of solving the shortest-path problem is $O(m^2)$ and we solve it $O(m)$ times, the time needed to solve the single-operator OFJS-S problem is within $O(m^3)$.

Next, we show that the one-operator case of the OFJS-WS problem is also solvable in polynomial time if the number of shift splits is at most $k$, although the run time grows exponentially in $k$.

**Lemma 2.** If there is a limit $k$ on the number of shift splits then the one-operator case of the OFJS-WS problem can be solved in $O(m^{2k+3})$ time.

**Proof.** It is easy to see that the best choice of a split would be such that the operator's split (idle period) would start at the end time of one job and end at the start time of another job. Thus, we have periods of continuous working time spaced by idle periods. With at most $k$ splits, there are at most $2k$ jobs that form the bookends of idle periods. There are at most

$$\binom{m}{2k} = O(m^{2k})$$

ways in which this can be done. This is an upper bound of the number of possible shifts and it may include invalid splits. An invalid split occurs, for example, when the end time of the first job is greater than the start time of the second job. Because it is difficult to find the precise number of valid splits, we use an upper bound in our arguments.

With $m$ jobs there are $O(m)$ possible combinations of operator shift start times. For each combination of shift start time and the choice of $2k$ idle-period bookend jobs, we can obtain the best assignment by solving a shortest-path problem, which requires $O(m^2)$ steps. This means that overall, the one-operator case of the OFJS-WS problem with bounded number of shift splits can be solved in at most $O(m^{2k+3})$ steps. Hence proved.

Next, we provide the pseudo-code for an algorithm that can be used to solve the one-operator case of OFJS-WS with at most $k$ shift splits. This algorithm is used as part of Heuristics $A_2$ and $A_3$ in Section 3.

**Algorithm for solving $k$-split one-operator OFJS-WS**

1. for $j = 1$ to $m$:
   2. consider $s_j$ as the start of the spread and $s_j + s$ as the end of the spread;
3. for all possible $k$ pairs of jobs;
4. each pair of jobs $j_1, j_2$ determines a possible split: if $e_{j_1} < s_{j_2}$ then it defines a split $(e_{j_1}, s_{j_2})$; if $e_{j_1} \geq s_{j_2}$ then it does not define a split;
5. if splits overlap, then combine them into one (large) split;
6. if a split exceeds the spread (either starts earlier than $s_j$ or ends later than $s_j + s$), then cut the split such that it lies entirely within the spread;
7. calculate the total duration of splits. If the total duration exceeds $s - w$, then sort the splits by start time,
and in this sequence keep as many splits as possible while ensuring that the total duration of splits does not exceed $s - w$. Use $d_i$ to denote the total duration of the selected splits;
8. adjust the end of the spread such that the end equals $s_f + w + d_i$;
9. the spread is cut into at most $(k + 1)$ segments. Consider each segment as a single spread and solve the one-operator OFJS-S on each segment. Combine these solutions to obtain the optimal assignment for the current spread with current splits. When a better assignment is found, keep that as the current best solution;
10. when all possible spreads and splits have been evaluated, report the best solution.

3. Heuristics

Given that the OFJS-WS problem is NP-hard, it is natural to spend effort on developing approximate solution techniques. In what follows, we develop three heuristics for solving the OFJS-WS problem, which are labeled as $A_1$, $A_2$, and $A_3$. In all three cases, we are interested in three characteristics of the algorithms, namely: speed, worst-case performance, and average performance. The goodness of a heuristic is often measured by its approximation ratio (Hochbaum, 1997). For the sake of completeness, we include a formal definition of the approximation ratio in the following paragraph. Because algorithms with a good approximation ratio do not sometimes have a good average performance, we develop multiple heuristics and experiment with real data to understand the speed and average and worst-case performance trade-offs implied by each algorithm.

Approximation ratio: Different versions of approximation ratios are used widely. We use the reciprocal of the definition given in Arora and Lund (1997, p. 400). That is, an algorithm achieves an approximation ratio $\rho$ for a maximization problem if, for every instance, it produces a solution of value at least $\rho \times \text{OPT}$, where $\text{OPT}$ is the value of the optimal solution.

3.1. The greedy approach ($A_1$)

Martello and Toth (1986) and Fischetti et al. (1992) both introduce algorithms based on the idea of assigning jobs to operators in a greedy fashion. Jobs are sorted according to some rule (e.g., by duration or by start time) and then assigned to available operators in that order.

Our greedy algorithm sorts jobs by duration and then assigns them in this sequence. The first job assignment always activates a new operator. For subsequent jobs, whenever a job cannot be assigned to one of the operators that has been previously activated, the algorithm activates a new operator. If all $n$ operators are active, and the job cannot be assigned to any available operator, then it is performed using overtime. If a job can be assigned to multiple operators, there are many possible ways to break the resulting tie. Our algorithm breaks the tie by assigning such jobs to operators who were activated the earliest. A pseudo-code for the greedy algorithm is shown below.

Pseudo-code for $A_1$: Let $P_u$, $P_a$, and $P_f$ denote, respectively, the set of unassigned operators, active operators, and full operators. Operators whose shifts are fully utilized are called full operators. A greedy algorithm can be constructed as follows.

$A_1$: The Greedy Heuristic

1. sort jobs according to the chosen criterion;
2. set $j = 1$ and $i = 0$;
3. while $j \leq m$ and $i < n$;
4. search for the set $P \subset P_a$ of active operators that can cover $\epsilon$;
5. if $P = \emptyset$ and $P_u \neq \emptyset$, then set $i = i + 1$ and assign job $j$ to operator $i$;
6. update $P_u$, $P_a$, and $P_f$;
7. set $j = j + 1$;
8. end

Approximation ratio: The example below shows that the greedy algorithm’s approximation ratio can be arbitrarily close to zero.

Example 1. Consider an instance of the OFJS-WS problem with one operator, $w = (m - 1)d$, and $s = md$, where $m$ is the number of jobs and $d$ is the duration of jobs $2, 3, \ldots, m$. Job 1’s duration is $d + \epsilon$, and the job start time is as follows: $s_1 = 0$, $e_1 = d + \epsilon$, $s_2 = md$, $e_j = s_j + d$, and $s_{j+1} = s_j + d$, for $j = 2, 3, \ldots, m$. A graphical representation of this problem (with $m = 5$) is shown in Fig. 2.

The optimal assignment to one operator would consist of jobs $2, 3, \ldots, m$. However, if we sort jobs by duration we would assign job 1 to the operator and cannot thereafter assign any other jobs because all other jobs violate the spread-time constraint. Hence, the greedy algorithm assigns $d + \epsilon$ units of work, whereas the optimal value is $(m - 1)d$. Then,

![Fig. 2. A graphical representation of job start and end times in Example 1.](image-url)
the approximation ratio is \((d + \epsilon)/(m - 1)d\), which goes to zero as \(m \to \infty\) and \(\epsilon \to 0\). The same argument would also apply if we were to sort jobs by start times, because job 1 had the earliest start time.

### 3.2. The Two-Stage Approach (A2)

As the name suggests, the two-stage heuristic solves the OFJS-WS problem in two stages. The first stage determines the report times of operators, whereas the second stage consists of an iterative upper and lower bounding approach that assigns jobs to operators whose report times have been fixed. We use the best lower bound as the heuristic solution. The successive bounding approach serves to improve the quality of the solution.

The first stage uses the solution of a polynomially solvable relaxation of the OFJS-S problem. This relaxation is called the preemptive and partial-credit version of the OFJS-S, which we denote in this paper as the PP-OJFS-S. Preemption means that jobs may be divided into several parts and each part may be covered by a different operator. Partial credit indicates the setting in which credit is applied even when a job is only partially covered. For example, if partial credit were granted, we would be able to divide a 2-hour job into two pieces of 1 hour each, cover a 1-hour piece without covering the remainder, and still count that as finishing 1 hour of productive work. We use the PP-OJFS-S problem rather than the PP-OJFS-WS problem to obtain report times because the former can be solved in polynomial time. In contrast, we are not able to show that the PP-OJFS-WS problem can be solved in polynomial time even with limited shift splits.

Report times are assumed to be known in the second stage, in which we assign jobs to operators. Note that the second-stage problem is still NP-hard (Kolen and Kroon, 1993). In that stage, we develop an iterative approach for solving the OFJS-WS problem with fixed report times and limited shift splits. This approach is based on the well-known subgradient optimization procedure (see, for example, Fisher (1981) and Kroon et al. (1995)). In what follows, we describe the two stages in detail.

**The First Stage:** Because the PP-OJFS-S problem is central to the first stage, we begin by formulating the PP-OJFS-S problem and then show that it is solvable in polynomial time. For any instance of the OFJS-S problem, we first define \(\delta_t\) to be the stacking of pieces of work at each \(t\in \{0, \ldots, t_{\text{max}}\}\). In particular, \(\delta_t = \{|j: s_j \leq t \leq e_j\}\) is the total number of jobs that need to be covered at time \(t\). Then we define \(S := \{s_j : j \in J\}\) as the set of job start times and \(T := S \cup (S + s) \cup \{e_j : j \in J\}\) as the set of critical time points. Clearly, \(S\) is the set of all possible operator report times, and \(T\) contains all job start and end times and all possible start and end times of operator shifts.

Figure 3 is a depiction of how the critical times are obtained. In this diagram, \(t_1\) to \(t_4\) are jobs’ start and end times, so they are automatically critical times. In addition, \(t_5 = t_1 + s\) and \(t_6 = t_2 + s\) are two additional critical times. We assume that the time points in \(T\) are sorted chronologically. Let \(p = |T|\). Then we can write \(T = \{t_k\}_{k=1}^p\) where \(t_k < t_{k+1}\). Moreover, for any \(t \in T\) we define \(c_t := t_{k+1} - t_k\) \((c_t := 0)\) as the weight of time interval \([t_k, t_{k+1}]\).

Let \(y_t\) be the number of operators who begin their shifts at time \(t\), for \(t \in S\). Because there are \(n\) operators to schedule, we must have \(\sum_{t \in S} y_t \leq n\). Let \(n_t\) denote the number of operators who are on duty during \([t, t + 1]\). Then, \(n_t\) should count all of the operators whose report times are within \([t - s + 1, t]\), or \(\tau : \tau \leq t \leq \tau + s - 1\), which can be written as \(n_t = \sum_{t \in S; t \leq \tau \leq \tau + s - 1} y_t\). Next, we define \(x_t\) as the total number of jobs that can be covered at each \(t \in T\). With preemptive assignment and partial credit, \(x_t\) and \(y_t\) are related as follows: \(x_t = \min(n_t, \delta_t)\), and the PP-OJFS-S problem can be formulated as shown in Equations (6) to (11).

\[
\max_{\{x_t, y_t\}} \sum_{t \in T} c_t x_t, \quad (6)
\]

subject to:

\[
x_t - \sum_{t \in S: t \leq \tau \leq \tau + s - 1} y_t \leq 0, \quad (7)
\]

\[
x_t \leq \delta_t, \quad t \in T, \quad (8)
\]

\[
\sum_{t \in S} y_t \leq n, \quad (9)
\]

\[
x_t \geq 0, \quad \text{integer}, \quad t \in T, \quad (10)
\]

\[
y_t \geq 0, \quad \text{integer}, \quad t \in S. \quad (11)
\]

**Lemma 1.** Preemptive and partial credit version of the OFJS-S problem is polynomially solvable.

**Proof.** We prove Lemma 1 by arguing that the coefficient matrix of constraints in the formulation shown in (6)–(11) is totally unimodular. Let \(x\) denote a vector of \(x_t\)s and \(y\) denote a vector of \(y_t\)s. Then, the constraints in (7)–(11) can be written as follows:

\[
A \begin{bmatrix} x \\ y \end{bmatrix} \leq b
\]

\[
x \in \mathbb{Z}_+^{|T|}
\]

\[
y \in \mathbb{Z}_+^{|S|}
\]
The coefficient matrix $A$ and the column vector $b$ have the following structures:

$$
A = \begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22} \\
A_{31} & A_{32}
\end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix}
b_1 \\
b_2 \\
b_3
\end{pmatrix},
$$

where

$$
A_{11} = A_{21} = \begin{pmatrix}
1 \\
. \\
. \\
. \\
1
\end{pmatrix}_{|T| \times |S|}
$$

$$
A_{22} = (0)_{|T| \times |S|}, \quad A_{31} = (0 \ldots 0)_{|T| \times |L|}, \quad A_{32} = (1 \ldots 1)_{|T| \times |L|},
$$

$$
b_1 = [0, \ldots , 0]_{1 \times |T|}, \quad b_2 = \begin{pmatrix}
\delta_1, \ldots , \delta_{\text{num}(1)}}_1 \|_{|T|}
\end{pmatrix}, \quad \text{and} \quad b_3 = (n), \text{where } [.]^T \text{ indicates the transpose of a matrix.}
$$

It is easy to see that $A^T$ satisfies the condition in Theorem 5.23 in Korte and Vygen (2006, p. 103) for being a totally unimodular matrix. Note that Korte and Vygen (2006) state that this theorem was originally proved in Ghouila-Houri (1962). We include the theorem below and show how $A^T$ satisfies the key condition.

Theorem 1. (From Korte and Vygen (2006, p. 103)) A matrix $A = (a_{ij} \in \mathbb{Z}^{m \times n})$ is totally unimodular if and only if for every $R \subseteq \{1, \ldots , m\}$ there is a partition $R = R_1 \cup R_2$ such that $\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \in \{-1,0,1\}$ for all $j = 1, \ldots , n$.

We write the transpose of $A$ as follows:

$$A^T = \begin{pmatrix}
A_{11}^T & A_{12}^T & A_{13}^T \\
A_{21}^T & A_{22}^T & A_{23}^T \\
A_{31}^T & A_{32}^T & A_{33}^T
\end{pmatrix}.
$$

The lower part of $A^T$ is called the “interval matrix.” That is, each column contains either all zeros or one consecutive block of either all ones or all minus ones. Then, within a subset $R$ of $A$, the rows that belong to the lower part of $A^T$ contain consecutive blocks of ones or minus ones in each column. Therefore, we can partition rows in $R$ that belong to the lower part of $A^T$ such that the odd rows belong to $R_1$ and the even rows belong to $R_2$, to satisfy the condition $\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \in \{-1,0,1\}$ for every $j$. Moreover, because $A_{31}^T$ and $A_{32}^T$ are zero matrices and $A_{11}^T$ is an identity matrix, the rows in $R$ that belong to the upper part of $A^T$ can be assigned to $R_1$ and $R_2$ such that $\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \in \{-1,0,1\}$ remains intact for every $j$.

We have shown that $A^T$, hence also $A$, is totally unimodular. As a result, a linear relaxation of the PP-OJF-S problem has an integer solution and the problem is solvable in polynomial time by solving its linear programming-relaxation. This completes the proof. $\blacksquare$

At the end of the first stage, we obtain operator report times, which are used in the second stage. The job assignments obtained by solving the PP-OJF-S problem are ignored.

The second stage: Let $r = (r_1, \ldots , r_n)$ be the report times obtained from Stage 1 of the two-stage algorithm. Next, we present both upper and lower bounding approaches that can be iteratively improved. The upper bound is obtained by relaxing Constraints (2) within the OFJS-WS formulation presented in (1)–(5). We employ a set of non-negative multipliers $u = \{u_1, \ldots , u_n\}$, one for each constraint in Equation (2), resulting in the addition of the penalty term $\sum_{1 \leq j \leq m} u_j (\sum_{1 \leq i \leq n} x_{ij} - 1)$ to the objective function. Let $\underline{z}(u, r)$ denote the optimal value of the objective function given penalty terms $u$ and report time vector $r$. Then, a formulation of the Lagrangian relaxation of the OFJS-WS problem, which we denote by LR-OJF-WS, is as follows:

$$
\underline{z}(u, r) = \max_{|x_{ij}|} \sum_{1 \leq j \leq m} (d_j - u_j) \sum_{1 \leq i \leq n} x_{ij} + \sum_{1 \leq j \leq m} u_j,
$$

subject to

$$
x_{ij} + x_{ik} \leq 1, \quad j = 1, \ldots , m - 1, \quad i = 1, \ldots , n, \quad k \in I_j,
$$

$$
\sum_{1 \leq j \leq m} d_j x_{ij} \leq w, \quad i = 1, \ldots , n,
$$

$$
x_{ij} = 0, \quad j \in R_i(s) \quad i = 1, \ldots , n
$$

$$
x_{ij} \in \{0, 1\}, \quad i = 1, \ldots , n, \quad j = 1, \ldots , m.
$$

The terms $R_i(s)$ in Constraints (15) denote the sets of jobs that cannot be assigned to operator $i$ given that operator’s report time selected in Stage 1 and spread $s$. Specifically, given report time $r_i$ from the first stage, $R_i(s) = \{j: s_j < r_i \text{ or } e_j > r_i + s\}$. Note that the set $I_j$ in Equation (13) is slightly different from the set $I_j(s)$ in (3) because only the indices of overlapping jobs are included in $I_j$. Spread-time violations are avoided via Constraints (15). For this reason, we no longer show $s$ as an argument of $I_j$ in Equation (13).

Upon examining the LR-OJF-WS formulation, we observe that it can be decomposed into $n$ independent
assignment problems, one for each operator \( i \). Therefore, the LR-OFJS-WS problem can be solved by repeatedly solving one-operator problems, each of which is polynomially solvable when the number of permissible shift splits is finite (see Lemma 2). Before describing additional details of this approach, we provide an intuitive explanation behind the decomposition that results from the LR-OFJS-WS formulation.

Constraints (2) guarantee that a job would not be assigned to more than one operator. By relaxing Constraints (2), it becomes feasible in the LR-OFJS-WS formulation to make such assignments. In other words, after some jobs are assigned to one or more operators, we are still able to choose jobs from the original set of jobs for the remaining operators. This immediately means that optimal assignment to each operator is independent of other operators' assignments and the problem decomposes into independent problems for each operator. Each assignment of job \( j \) improves the value of the objective function by \((d_j - u_j)\), which can be affected by changing \( u_j \). In particular, a higher value of \( u_j \) makes it less desirable to multiply assign job \( j \) to several operators. This observation is the basis of a procedure for updating \( u_j \) that can iteratively improve the upper bound obtained via the LR-OFJS-WS formulation.

Recall from Equation (1) that the optimal value of the OFJS-WS problem is denoted by \( z \). It is straightforward to argue that \( z(u, r) \geq z \) for any \( u \geq 0 \) (Fisher, 1981). This means that solving the LR-OFJS-WS problem by decomposition is a polynomial-time upper-bounding procedure for the OFJS-WS problem when the number of allowable shift splits is bounded. Next, we present a lower bounding formulation, which is used for updating \( u \) iteratively in an attempt to improve our solution.

The lower bounding algorithm assigns jobs to each operator one at a time such that jobs assigned in an earlier position is a polynomial-time upper-bounding procedure. At the end of each iteration, the lower bound obtained is compared with the previous best lower bound (largest value of lower bounds obtained in previous iterations) and the new value is kept if it is higher, or else the previous best value is retained. We use the notation \( z(u, r) \) to denote the lower bound obtained at an arbitrary iteration.

Next, we describe the two-stage algorithm below. We use \( k \) as iteration count and attach superscript \((k)\) to each term to denote iteration number. The maximum number of iterations is denoted by \( N \). The algorithm stops when \([z(u, r) - z(u, r)]/z(u, r)\) either reaches or drops below a predetermined threshold denoted by \( \delta \) or the maximum number of iterations \( N \) is exhausted.

**A2: The Two-Stage Heuristic**

1. (First stage) solve the PP-OJFS-S with spread limit \( s \); obtain a set of operator report times;
2. (Second stage) set the weight of each job equal to duration; set iteration count \( k = 0 \) and set \( u_j^{(k)} = 0 \);
3. while \([z(u, r) - z(u, r)]/z(u, r) > \delta \) and \( k \leq N \):
   4. execute the upper-bounding procedure and record the total assigned value \( z(u, r) \) and assignments \( x_j^{(k)} \);
   5. execute the lower-bounding procedure and record the highest lower bound at iteration \( k \) denoted by \( z(u, r) \);
   6. set \( k = k + 1 \);
7. end
8. report \( x_j \)'s that correspond to the best lower bound.

**Multiplier updating procedure:** The key to the second-stage approximation algorithm is the choice of the Lagrangian multipliers \( u \). Although it has not been proved that there exist \( u \) such that \( z(u, r) = z \), the Lagrangian relaxation method is widely used and performs well for many problem categories (Fisher, 1981). Because \( z(u, r) \) is a piece-wise linear function of \( u \) and one of its subgradients is known to be the vector \((\sum_j x_{ij} - 1, \ldots, \sum_j x_{im} - 1)\), we are able to utilize a commonly used approach for the multiplier updating, which is described next.

Let \( u_j = \max(0, u_j + \lambda(\sum_i x_{ij} - 1)) \), where \( x_{ij} \) is a solution of the upper-bounding procedure, and \( \lambda = \mu(z(u, r) - z(u, r))/\sum_j(\sum_i x_{ij} - 1)^2 \). In the previous expression, \( \mu \) is referred to as the step-size parameter. When implementing a two-stage algorithm, \( \mu \) is set equal to an initial value (we use a value of one) and its value is halved each time when the gap between \( z(u, r) \) and \( z(u, r) \) does not decrease after carrying out a certain number of iterations (we use a value of five).

**Approximation ratio:** We were unable to establish an approximation ratio for the two-stage heuristic. However, the successive bounding approach is expected (although not guaranteed) to improve the solution at each iteration, resulting in good overall performance. In the upper-bounding procedure, each job is allowed to be assigned multiple times. If this happens, the value \((\sum_i x_{ij} - 1)\) will be positive and the multiplier updating procedure will increase \( u_j \), which will cause a decrease in the value \((d_j - u_j)\). This makes job \( j \) less attractive in future iterations of the algorithm in both upper-bounding and lower-bounding procedures. Conversely, if job \( j \) is not assigned, then its value \((d_j - u_j)\) will increase and the job will become a more attractive candidate for assignment in the next iteration. The updating step presents an opportunity to reduce the value of the upper bound and to increase the value of the lower bound at each iteration. For these reasons, the two-stage algorithm
performed well in numerical experiments with real data (see Section 4).

### 3.3. The decomposition-based approach (A3)

The third approach we propose uses a decomposition to solve the OFJS-WS problems. The idea is to decompose the $n$-operator assignment problem into $n$ separate one-operator cases. We assume a limited split setting. Recall that such problems can be solved in polynomial time, as shown in Section 2.1. At each iteration we introduce a new operator and obtain the best assignment with the remaining jobs. In the following description of $A_3$ we use $\tilde{J}$ to denote the set of unassigned jobs and $i$ to denote the current operator being scheduled.

#### $A_3$: The Decomposition Heuristic

1. set $\tilde{J} = J$ and $i = 0$;
2. while $\tilde{J} \neq \emptyset$ and $i < n$
3. introduce a new operator;
4. maximize assignment to the operator by using limited split one-operator algorithm (see Section 2.1);
5. update $\tilde{J}$ and set $i \leftarrow i + 1$;
6. end

**Algorithm complexity:** Because the limited split one-operator algorithm runs in $O(m^{2k+3})$ time (see Lemma 2), it is clear that Algorithm $A_3$ also runs in $O(nm^{2k+3})$ time. It is polynomial in $m$ and $n$, but its run time increases exponentially in $k$.

**Approximation ratio:** Finding the exact approximation ratio of a decomposition algorithm for solving the FJS class of problems is difficult. For example, Fischetti et al. (1992) attempt to do so for the TFJS-S but did not find a precise approximation ratio. Instead, Fischetti et al. (1992) present an instance of the TFJS-S problem and use its solution to argue that the approximation ratio could not be larger than $3/2$. Recall that the TFJS-S is a minimization problem and therefore the approximation ratio is never smaller than one, and a higher approximation ratio implies worse performance in that case. Similar to these earlier papers, we are also unable to establish a precise approximation ratio for the OFJS-WS problem. Instead, we define $f(p) = 1 - [(p - 1)/p]^p$ and show that the approximation ratio lies in $[f(\infty), f(3)]$ with $f(\infty) = 1 - 1/e$ and $f(3) = 19/27$.

To establish the main result in this section, we begin by arguing that the OFJS-WS problem can be viewed as a special case of the Weighted Maximum Coverage Problem (W-MCP) (Khuller et al., 1999). An arbitrary instance of the W-MCP has a finite universe set $\Omega$, a positive weight $w_j$ for each element in $\Omega$, a collection $A = \{S_1, S_2, \ldots, S_k\} \subset 2^\Omega$ of $k$ subsets of $\Omega$, and a designated number $n$. The objective of the W-MCP is to find a subcollection $A'$ with cardinality at most $n$, such that the number of elements covered by $A'$ is maximized. Below, we present a greedy algorithm for solving W-MCP due to Cornuejols et al. (1977), which is known to have an approximation ratio of at least $(1 - 1/e)$.

#### Greedy Algorithm for W-MCP

1. Repeat $n$ times;
2. find one element $S'$ in $A$ that yields the largest increment of the current objective function if added to the solution;
3. add $S'$ to the solution, and delete $S'$ from $A$.

In the following proposition we establish the correspondence between the OFJS-WS and the W-MCP approaches and argue that $A_3$ is a version of the greedy algorithm for solving the W-MCP. From Cornuejols et al. (1977), it then follows that $A_3$ has an approximation ratio of at least $(1 - 1/e)$.

**Proposition 1.** $\rho(A_3) \geq 1 - 1/e$.

**Proof.** We perform the following transformation from the OFJS-WS formulation to the W-MCP formulation. Let the set of jobs be the universal set $\Omega$ and $d_j$ be the weight of the $j$th element, for $j = 1, \ldots, m$. Define the candidate set $J = \{S_1, S_2, \ldots, S_k\}$, in which $S_i$ is a set of jobs that can be assigned to an operator, and $\ell$ is the total number of possible sets. Let the number of operators $n$ be the number of subsets we can choose. Given this setup, the correspondence between the W-MCP and the OFJS-WS approaches is straightforward.

Note that the set of all possible one-operator assignments $J$ is determined by jobs’ start and end times, $w, s$, and the number of splits allowed. Therefore, it is difficult to calculate $\ell$ without obtaining all feasible assignments to a single-operator problem. Fortunately, it is not necessary to know all elements of $J$ to implement a greedy approach in our setting. If we were to select subsets $S_j$ in a greedy way in the W-MCP, then that would be equivalent to finding the best single-operator assignments from remaining jobs in $J$ at each assignment step. This can be accomplished in polynomial time when the number of splits is finite (see Lemma 2). This means that $A_3$ is equivalent to the greedy algorithm in Cornuejols et al. (1977). Therefore, from Cornuejols et al. (1977), we immediately have that $\rho(A_3) \geq 1 - 1/e$. ■

In order to establish a good upper bound, we need to construct examples whose approximation ratio is as close as possible to the lower bound established in Proposition 1. Note that the lower bound can be viewed as the limiting case of a sequence $f(p) = 1 - [(p - 1)/p]^p$, $p \geq 1$, in the limit as $p \to \infty$. If we find an example with performance $f(\hat{p})$ for some $\hat{p}$, then that establishes a good upper bound for all $p \leq \hat{p}$ because $f(p)$ is a decreasing sequence. If we find such examples for every $p$, then the lower bound is tight.

In what follows, we describe an example with approximation ratio $f(p)$ for $p = 3$; i.e., the approximation ratio is $f(3) = 19/27$. Note that $p$ is not a parameter of this example. In particular, it does not relate to either the number...
of jobs or the number of operators. Unfortunately, the approach used to construct this class of examples does not extend to cases with \( p > 3 \). Therefore, we were not able to find even sharper bounds on the approximation ratio of the decomposition algorithm proposed here.

**Proposition 2.** \( \rho(A_3) \leq 19/27 \).

**Proof 2.** We construct an example with three operators, \( s = w = 9 \), and nine jobs. The nine jobs are divided into three groups, each including three jobs. The start and end times of each job in group 1 are, respectively, \((0, 3 - \epsilon), (3 - \epsilon, 5), \) and \((5 + 4\epsilon, 9)\) with durations \(3 - \epsilon, 2 + \epsilon, \) and \(4 - 4\epsilon\). Jobs in groups 2 and 3 have the same sequence of durations, except that the start times are shifted by \(3 - \epsilon\) for group 2 and \(2(3 - \epsilon)\) for group 3.

The jobs are shown in Fig. 4, in which each row represents one group of jobs. Given this layout, it is easy to see that an optimal solution is to have each operator cover a group of jobs.

Next, consider what would happen if the decomposition algorithm is used to obtain the best solution for the first operator. With that approach, the optimal one-operator solution will be to assign all three jobs with duration \(3 - \epsilon\) to the first operator. The remaining jobs available for assignment to the second operator appear as shown in Fig. 5. Note that two jobs of duration \(2 + \epsilon\) cannot be combined with a job with duration \(4 - 4\epsilon\) because of the spread-time constraint. Therefore, at this step, the best one-operator assignment is to assign the three jobs with duration \(2 + \epsilon\) to operator 2. Finally, at the last step only the longest jobs remain. Since any two of the longest jobs cannot be assigned to one operator because of overlap or spread-time violation, the third operator only covers a single job of duration \(4 - 4\epsilon\).

Combining the results from the above steps, we see that \(A_3\) returns an objective value of \(19 - 8\epsilon\), whereas the optimal value is \(27 - 12\epsilon\). That is, the approximation ratio can be made arbitrarily close to \(1 - (2/3)^3\) via the choice of \(\epsilon\). This completes the proof of the Proposition.

### 4. Numerical experiments

We received extraboard operations data for five randomly picked months from a large transit agency that served as a research partner for this study. The agency has five garages and each month’s data came from a different garage. The data included all jobs that were assigned to either extraboard operators or on an overtime basis. The agency rarely dropped service. In fact, there were no examples of dropped service in our data set, which meant that the jobs in our data set represented the entire demand for extraboard services. From the data, we identified the subset of jobs that were known to be open a day before each day of operations and calculated the number of extraboard operators available to serve those jobs. Recall that a certain number of operators are placed on call duty each day; i.e., they are only assigned jobs that arise during the course of the day. We did not include those operators in our study.

Because extraboard operators’ wages are sunk, the transit agency attempts to assign as much work as possible in their regular work time. Our algorithms mimic this objective. The transit agency also considers overtime availability constraints on certain days of the year; e.g., Christmas Day when overtime availability is limited. Our algorithms ignore overtime availability constraints. Thus, they are close approximation of the problem faced by the dispatcher on most days of the year.

For each given day, the data sometimes contained one or more 8-hour-long jobs. We excluded such jobs from the data and reduced the number of available operators on that day by the same amount. This was done because assigning each 8-hour-long job to a single operator is trivially an optimal strategy, irrespective of other assignments. The set of jobs we worked with were the jobs that were left after this process of elimination. On many weekends and some weekdays, the number of individual 8-hour-long jobs exceeded the number of extraboard operators available to perform such duties. We excluded such days from our experiments altogether, which left 100 instances of the problem from the 5 months of data. In Table 2 we summarize the data. Empirical distributions of job start times and job durations, job numbers, and operator numbers are provided in Tables 1 to 4 in the Online Supplement.

Table 2 shows that there was a great deal of variation in open jobs from one day to another. The average job duration was different for each garage and lay between 3 and 4 hours. The job durations were quite variable—the
The purpose of the numerical experiments was to identify an algorithm that performed well relative to the CPLEX solution and that was also fast. We calculated three performance metrics for this purpose: two of these measured the relative quality of the solution produced and one measured speed. The metrics were (i) algorithm solution/CPLEX solution (in percent); (ii) percentage of times that the algorithm solution is $\geq 99\%$ of the CPLEX solution; and (iii) computation time (in seconds). The results are reported in Table 5. A quick look at this table reveals that $A_1$ runs quite fast but produces the worst average performance among the three algorithms. $A_2$ has much better average performance than $A_1$ but runs much slower. Algorithm $A_3$ produces good average performance and runs fast at the same time. Moreover, $A_3$ is the only algorithm that has a proven non-zero approximation ratio. Therefore, the experiments support the claim that algorithm $A_3$ is the best among the three algorithms evaluated.

To further compare the three algorithms, we developed complementary cumulative frequency plots of the number of minutes assigned by each method; see Fig. 6. Complementary cumulative frequency is $100\%$ minus the cumulative frequency of the assigned time for each algorithm. Roughly speaking, if an algorithm’s performance is to the right, then that implies a superior performance. We observe

<table>
<thead>
<tr>
<th>Garage</th>
<th>Min (%)</th>
<th>Max (%)</th>
<th>Avg (%)</th>
<th>Std. dev (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Garage 1 (21)</td>
<td>17.6</td>
<td>62.5</td>
<td>40.1</td>
<td>11.6</td>
</tr>
<tr>
<td>Garage 2 (20)</td>
<td>22.2</td>
<td>73.7</td>
<td>45.0</td>
<td>14.9</td>
</tr>
<tr>
<td>Garage 3 (21)</td>
<td>16.7</td>
<td>85.7</td>
<td>51.3</td>
<td>16.2</td>
</tr>
<tr>
<td>Garage 4 (19)</td>
<td>18.5</td>
<td>66.7</td>
<td>44.8</td>
<td>15.5</td>
</tr>
<tr>
<td>Garage 5 (19)</td>
<td>16.7</td>
<td>62.5</td>
<td>33.9</td>
<td>12.2</td>
</tr>
</tbody>
</table>
Table 5. Comparison of the algorithms’ performances

<table>
<thead>
<tr>
<th>Metric</th>
<th>Garage</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution from algorithm (%)</td>
<td>1</td>
<td>89</td>
<td>92</td>
<td>98</td>
</tr>
<tr>
<td>Solution from CPLEX (%)</td>
<td>2</td>
<td>96</td>
<td>95</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>94</td>
<td>92</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>96</td>
<td>97</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>92</td>
<td>97</td>
<td>99</td>
</tr>
<tr>
<td>Frequency that algorithm returns ≥ 99% of CPLEX solution (%)</td>
<td>1</td>
<td>6</td>
<td>11</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>23</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6</td>
<td>39</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0</td>
<td>45</td>
<td>86</td>
</tr>
<tr>
<td>Computation time (Second)</td>
<td>1</td>
<td>2.1</td>
<td>240</td>
<td>3.4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.4</td>
<td>320</td>
<td>4.6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1.7</td>
<td>120</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.0</td>
<td>170</td>
<td>3.3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>2.7</td>
<td>290</td>
<td>5.0</td>
</tr>
</tbody>
</table>

in Fig. 6 that $A_3$ is to the right of $A_1$ and $A_2$ for nearly all values of assigned work. For example, we draw a vertical line in Fig. 6 at 4000 minutes to draw attention to the fact that whereas $A_1$ and $A_2$ assign more than 4000 minutes of work in about 20% of all problem instances, $A_3$ does so in nearly 34% of instances. Therefore, the performance of $A_3$ dominates the performance of the other two heuristics in the usual stochastic order. A random variable $X$ is stochastically smaller than another random variable $Y$ in the usual order, written $X \leq_{ss} Y$, if $E[\phi(X)] \leq E[\phi(Y)]$ for all non-decreasing functions $\phi$ for which the expectations exist (see Shaked and Shanthikumar (1994) and Müller and Stoyan (2002) for further details).

Algorithm $A_3$ runs fast and does not require the transit agency to invest resources in purchasing a commercial optimization software such as CPLEX. The data also show that the use of $A_3$ could save somewhere between 1.2 to 6.5 hours of overtime on weekdays. The average savings per day per garage is 3.6 hours. Using an average overtime wage rate of $42, which we obtained from our research partner, this implies approximate annual savings of $196,560 (which is obtained by calculating $5 \times 52 \times 5 \times 3.6 \times 42$).

5. Concluding remarks

The paper is motivated by extraboard operator scheduling and work assignment problems that are faced by transit agencies on a daily basis. We present a model and three algorithms for solving the operational fixed job scheduling problem with work-time and spread-time constraints (OFJS-WS). We show that the OFJS-WS problem is NP-hard. We prove that $A_3$, a decomposition-based approach, has an approximation ratio that lies in the range $[1 - 1/e, 19/27]$. We perform numerical experiments using data from the collaborating transit agency and show that our algorithm provides close-to-optimal solutions and has the potential to improve extraboard work assignments. Ongoing efforts by the authors are being focused on solving the day-of scheduling problems and improving understanding of the relationship between the day-before scheduling and the day-of scheduling problems.

Acknowledgements

The authors are also grateful to the IIE Transactions’ review team and Departmental Editor Professor Sanjay Mehrotra for helping us improve the paper.

Funding

This material is based upon work supported in part by grants from the Center for Urban and Regional Affairs’ Faculty Interactive Research Program, the Center for Transportation Studies, and the National Science Foundation under grant no. CMMI-1332680.

Supplemental material

Supplemental data for this paper can be accessed on the publishers website.

References


**Biographies**

Fei Li is a Ph.D. candidate in the Industrial & System Engineering Department of the University of Minnesota. His research focuses on algorithms for solving combinatorial optimization problems.

Diwakar Gupta is a Professor of Industrial & Systems Engineering at the University of Minnesota. He also holds a courtesy appointment as an affiliate senior member in the Health Services Research, Policy, and Administration Division of the School of Public Health. He earned a Ph.D. in Management Sciences from the University of Waterloo. His research focuses on healthcare delivery systems, state transportation agencies’ operations, and supply chain and revenue management. His research has been funded by a variety of federal and state agencies (e.g., DHHS, NSF, AHRQ, VHA, Mn/DOT, NSERC, SSHRC, and CHSRF), as well as companies, and his papers have appeared in all major journals in the field of Operations Research/Management. He was elected in 2011 and 2013 to be the chair of the Healthcare Operations Management Special Interest Group of the Manufacturing and Service Operations Management Society. More information about his research projects can be found by visiting the web page of his research lab, Supply Chain and Operations Research Laboratory, at www.isye.umn.edu/labs/sorlab.