Single-machine scheduling with stepwise tardiness costs and release times

Güvenç Şahin[*] and Ravindra K. Ahuja[†]

March 10, 2011

Do not cite this article without a permission from the corresponding author.

Abstract

We study a scheduling problem that belongs to the yard operations component of the railroad planning problems, namely the hump sequencing problem. The scheduling problem is characterized as a single-machine problem with stepwise tardiness cost objectives. This is a new scheduling criterion which is also relevant in the context of traditional machine scheduling problems. We produce complexity results that characterize some cases of the problem as pseudo-polynomially solvable. For the difficult-to-solve cases of the problem, we develop mathematical programming formulations, and propose heuristic algorithms. We test the formulations and heuristic algorithms on randomly generated single-machine scheduling problems and real-life data sets for the hump sequencing problem. Our experiments show promising results for both sets of problems.

Keywords: single-machine scheduling, stepwise tardiness, hump sequencing, railyards, yard operations.

1 Introduction and motivation

Many researchers have studied tardiness type scheduling criteria in machine scheduling applications in the past. Naturally, the tardiness criterion is valid only when there is a deadline associated with the completion of the jobs. The traditional versions of this type of criterion are known to minimize

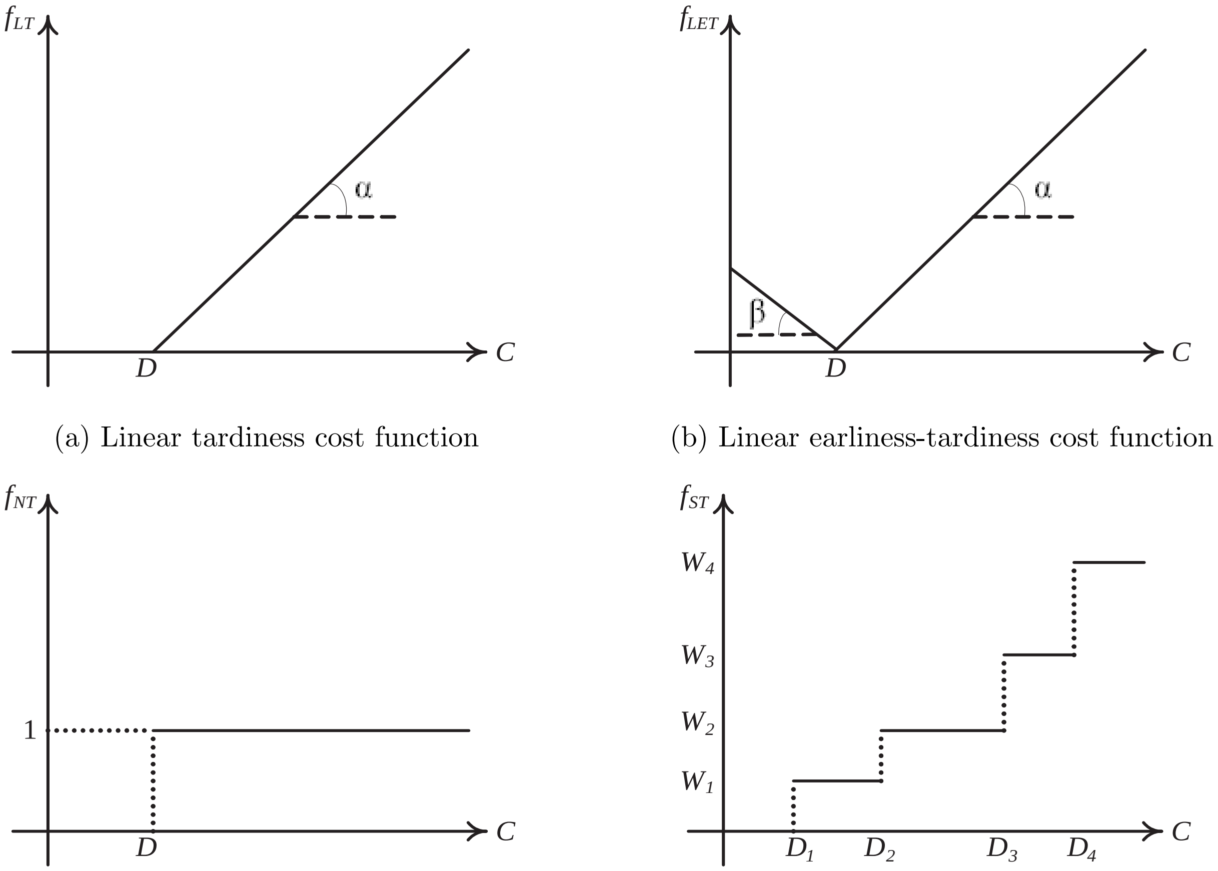
- the sum of linear tardiness costs of jobs,
- the sum of linear earliness and tardiness costs of jobs,
- the number of tardy jobs, and

[*] Corresponding author. Manufacturing Systems/Industrial Engineering, Faculty of Engineering and Natural Sciences, Sabanci University, Orhanli, Tuzla, 34956 Istanbul, Turkey. guvencs@sabanciuniv.edu.

[†] Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611 USA. ahuja@ufl.edu.

- the weighted versions of the above criteria.

In this study, we investigate a new machine scheduling criterion, which considers *stepwise tardiness* cost objective functions. For both the traditional versions of the tardiness cost functions and the stepwise tardiness cost function, see Figure 1.



(a) Linear tardiness cost function

(b) Linear earliness-tardiness cost function

(c) Number of tardy jobs function

(d) Stepwise tardiness cost function

Figure 1: Types of tardiness cost functions

The traditional linear tardiness cost function for a job is a linear function of job completion time:

$$f_{LT}(C) = \begin{cases} 0, & C \leq D \\ \alpha(C - D), & C > D. \end{cases}$$

where C is the completion time and D is the due date of the job. Accordingly, the objective function that considers both the earliness and tardiness of the job can be represented as

$$f_{LET}(C) = \begin{cases} \beta(D - C), & C \leq D \\ \alpha(C - D), & C > D. \end{cases}$$

In order to minimize the number of tardy jobs, each job is described by a cost function

$$f_{NT}(C) = \begin{cases} 0, & C \leq D \\ 1, & C > D. \end{cases}$$

Any of these cost functions can be considered as piecewise linear functions that are composed of at most two pieces. Their weighted versions simply multiply the non-zero valued pieces with a positive weight factor.

Accordingly, we describe the stepwise tardiness cost function for a job as follows:

$$f_{ST}(C) = \begin{cases} 0, & C \leq D_1 \\ W_1, & D_1 < C \leq D_2 \\ W_2, & D_2 < C \leq D_3 \\ \vdots & \vdots \\ W_{s-1}, & D_{s-1} < C \leq D_s \\ W_s, & D_s < C. \end{cases}$$

where the job is associated with a set of due dates $\{D_1, D_2, ..., D_{s-1}, D_s\}$ rather than a single due date, and each due date is associated with a constant tardiness cost/weight.

We have first observed this type of tardiness criterion in a very non-traditional scheduling context, namely the railroad yard operations. The particular task to be scheduled in this context is involved with disassembling the inbound trains arriving at a yard into individual cars (or groups of cars) that are later to be assembled (or connected) to outbound trains departing from the yard.

Before we further detail the particular scheduling problem in the yard operations context, we make the following observations that motivate the relevance of this tardiness criterion to more traditional scheduling environments:

Fixed transportation/shipment schedules. We consider a manufacturing environment where completed customer orders (jobs) need to be shipped in some mode of freight transportation. For several modes of freight transportation (including LTL and expedited ground), continuous service during the day (or shift) is not available. Rather, the transportation service is dispatched at some regular or non-regular periodic manner. Therefore, orders that are completed are likely to wait until the next transportation service is available (e.g., at the end of the day or at the end of the shift). All orders that have to wait for a particular transportation service are effectively completed at the same time, which is the time of the next (or earliest) transportation service. Although the actual completion time is still a continuous function, the tardiness of the orders change only at certain time epochs that correspond to transportation service dispatches. A similar environment is discussed in Curry and Peters (2004) for a parallel machine scheduling problem where the due dates correspond to pick-up cutoff times of transportation services.

Batch production. We consider a producer with a 'fixed product portfolio' (with no customization) for which economies of scale apply in production costs (i.e. the larger the size of the production batch for a particular product the more economical it is to produce). A typical customer order may contain a variety of product types with a common delivery date for all products or a set of preferable delivery dates different for each product type. When orders from several customers are received, orders for the same product type are combined into a single production batch. This consolidation, however, imposes several delivery dates (due to delivery dates imposed by different customers), and it is usually not possible to simultaneously satisfy all delivery dates. From a customer service perspective, delaying the completion time of this batch implies a decreasing satisfaction (or an increasing dissatisfaction) for some of the customers since the delivery date for some orders may be missed as time elapses. The value of this production batch from a service level perspective changes at certain time epochs that correspond to individual customer preferred delivery dates. This is a very typical case for a chemical producer that runs a single production batch for each product type during a fixed length planning horizon (e.g. a week). When customer orders are combined into production batches, each production batch has the type of tardiness cost structure described by $f_{ST}(C)$. The production planning decision is concerned with scheduling/sequencing the run of the single production batch of each product type during a week so that more customer satisfaction is attained.

In a seemingly unrelated reference, Birge and Maddox (1995) study the expected completion time analysis in project management context via a stochastic network model. Birge and Maddox, however, discuss different types of tardiness schemes as an introduction to justify the contribution of their study and applications in practical scheduling situations other than project management context. They discuss three situations to exemplify the cases where regular linear tardiness costs are indeed approximations of more realistic versions which are difficult to handle. After discussing the two rather traditional examples, they introduce another case as follows:

> *A third type of tardiness cost may occur when the deadline is actually a target date for the beginning of a final activity (such as shipping), which is assumed to consume some regular time (for example, due to a fixed transportation schedule), but which may be reduced at some cost that is proportional to the time reduction. While this cost is most likely to have some jumps (due to changing transportation modes, for instance) and to increase at a greater than linear rate with respect to tardiness, overlapping piecewise linear tardiness objectives at different deadlines (possibly for each mode) may yield good approximations.*(Birge and Maddox (1995), p. 838)

In relation to the fixed transportation schedule example, we also note that discussion regarding the *change in transportation mode* is very critical in this case. It is usually true that when an order misses the targeted transportation time, the cost of transportation may increase either due to a change in the mode of transportation or having to request a non-regular service to expedite the transportation process. Within the same framework we have discussed above, the cost of a tardy job to the manufacturer is usually represented in terms of the loss of goodwill (degrade in customer service), which lead to our stepwise version of the tardiness costs. To exemplify, a customer would definitely distinguish an on-time delivery from a 2-day late delivery as well as a 2-day late delivery from a 3-day late delivery, but would not care if the order is delivered in the morning or in the evening of the earlier (or later) date. On the other hand, the production schedule during the day is critical for the manufacturer since there are several other jobs to be scheduled.

In different cases we discuss above including the yard operations scheduling problem of our particular motivation, the stepwise tardiness costs of a job is an increasing function of the completion time as shown in Figure 1. Although there might be other particular cases where the stepwise tardiness weights are not necessarily increasing since they are dependent on a larger set of environmental parameters, we only focus on the cases where the tardiness costs only increase in a stepwise fashion. Our attempt should be considered as a starting point rather than an end; we believe that the problem introduced in this study has a practical relevance for many scheduling environments that may replace some of the traditional historically accepted scheduling criteria.

The contribution of this study is two-fold:

- We study a relatively new class of single-machine scheduling problems and develop solution methods for some difficult cases of this problem class.
- We apply the methods developed for the new scheduling problem to the hump sequencing problem of railroad yard operations; we solve real-life problems to illustrate the potential improvements in yard operations.

In the next section, we introduce the hump sequencing problem. Section 3 formally defines the single-machine scheduling problem with stepwise tardiness costs. We develop integer programming formulations for the problem and discuss some complexity results in Section 4. Section 5 and Section 6 present the heuristic algorithms and computational results, respectively.

2 Yard operations and hump sequencing problem

Railroad freight transportation is based on a consolidation plan of shipments in order to benefit from economies of scale in core transportation costs. Individual shipments are not necessarily shipped directly from their origins to their destinations; this would require a train service for each and every origin-destination pair. The consolidation plan that combines several shipments into groups of shipments is called a *blocking plan* and a consolidated set of shipment is known as a *block*. According to the blocking plan, a shipment starts its journey by being assembled into a block of railcars with several other shipments whose destinations are not necessarily the same as that of itself. This particular block is disassembled at the block's destination, and the shipment might be reassembled into another block. This process is called *classification*, and it can be performed only at those stations that are designated as *classification yards*. In other words, yards are analogous to airline hubs as shipments to passengers, and reclassification of a shipment from one block to another corresponds to a passenger's connecting from one flight to another.

Yards are crucial facilities of the railroad network; they are called the nerve centers of the railroad system. They are physically capacitated with respect to the number of cars that can be classified and with respect to number of blocks that can be made in a given time period. Therefore, yard capacities play a significant role in routing the shipments over the rail network. When its capacity is increased, a yard can make (process) more cars and blocks. In that case, some of the cars which cannot be originally routed through this yard due to the capacity restrictions, can be routed through this yard; the travel time of these cars from their origins to their destinations might be shortened. If the trains connecting shipments to each other cannot be processed in a timely manner, it results in either delaying trains for incoming shipments to catch their outgoing trains or shipments missing their outbound connections. Eventually, this will lead to delays in delivery of the shipments, which will disturb the service level.

Yard operations process contains in itself three major interdependent serial components: the hump sequencing, block-to-track assignment, and assembly sequencing. Each component can be considered as a separate planning problem that is the subproblem of the larger yard operations problem. The process of *hump sequencing* is the foremost operation. Inbound trains are accepted at the receiving yard as they arrive. The cars on the inbound trains need to be disassembled and positioned on the classification tracks; they are later to be assembled onto outbound trains awaiting at the departure yard. For the cars to be disassembled from inbound trains and positioned on the classification tracks, each inbound train is *humped* over an artificially built hill known as the *hump*, a special single lead track that uses the force of gravity to propel the cars down the track. There is usually only one hump at a typical classification yard; for safety purposes, the cars can go over this track at a limited speed, which makes humping the bottleneck operation at most yards. Since this is a critical operation, and there are several inbound trains at a given yard on a given day, humping of trains should be sequenced carefully so that more cars can catch their outbound trains, or fewer cars miss their outbound connections.

2.1 Hump sequencing problem

Hump sequencing can be done by assuming a given block-to-track assignment and pre-determined cut-off times for starting the assembly of outbound trains (set to a time before their departure). From a mathematical perspective, solving the hump sequencing problem alone provides an upper bound in terms of the respective objective function (minimize missing connections or throughput time for an average car) when the classification track assignments and assembly sequence is given. In practice, both these problems are usually considered as having predetermined solutions. Block-to-track assignment plans are made based on historic information and experience; they are not dynamic assignments that can change during a day or a shift. Assembly sequencing decisions have less degrees of freedom than the hump sequencing decisions since they are mostly imposed by the departure sequence of trains.

The hump sequencing problem is an instance of the machine scheduling problem with stepwise increasing tardiness costs when the objective is to maximize the realized car-train connections. We correspond the relation between an inbound train and the set of outbound trains, for which this inbound train contain some cars, to the relation between a job and the set of its due dates. Each outbound train implies a certain connection time based on its scheduled departure time, which corresponds to a due date. If the inbound train misses the connection time for a particular train, the cars from the inbound train connecting to the outbound train have to wait until the same outbound train service is repeated (which is usually a day later). As the humping of an inbound train is delayed, the number of cars that miss their connections increase in a stepwise fashion at those time instants imposed by the connection times of the outbound trains. The arrival times of the inbound trains correspond to job release times as the humping times (a function of the number of cars on the train and physical length of the train) correspond to job processing times. The number of cars in each connection set determines the size of stepwise jump in tardiness weights at time epochs that represent the due dates corresponding to the connection times of outbound trains. As a result, we have a one-to-one correspondence between a single-machine scheduling problem instance and a hump sequencing problem instance. In the rest of the study, we use a notation more relevant to the machine scheduling literature for ease of readability. Yet, we will come back to the hump sequencing problem in Section 6 with a computational study on a real-life data set.

2.2 Literature on yard operations problem

Yard operations have been studied both by academicians and practitioners in the past as it has always been evident that the yards are the bottlenecks of blocking plans. Past approaches, however, mainly focused on

- predicting the yard performance, and
- evaluating the criticality of resources.

The techniques are based on queuing models and statistical analysis tools (Petersen 1977a, Petersen 1977b, Martland 1982, Turnquist and Daskin 1982). The goal of these studies is to understand whether or not a yard can handle a given traffic pattern under a set of predetermined decisions and resource levels. None of these studies have considered tools to construct an operational plan in order to make decisions regarding operations scheduling and resource allocation.

The story is arguably different for the block-to-classification track assignment problem. This problem has always been considered as a challenge and assumed to be one of the most prominent aspects in determining the usable blocking capacity of a yard. For decades, the railroads has been handling this issue based on historic information and experience. Researchers, on the other hand, have approached this problem in order to find alternate assignment plans that significantly change the way classification tracks are assigned; such plans do not work in coordination with the current assignment scheme. The proposed plans do not necessarily subsume the traditional methods railroads have used. Therefore, they have never been implemented due to risk of dramatic changes in principles although they are well-known by railroads (see Daganzo *et al.* 1983, Daganzo 1986, Daganzo 1987a, Daganzo 1987b).

Recent works of Kraft are milestones in terms of yard planning tools. Kraft (2000) studies the hump sequencing problem with a real-time planning perspective. Kraft (2002a) and Kraft (2002b) study the car commitment and block-to-track assignment problems, respectively. The latter focuses on the assignment problem with guaranteed connections (i.e. cars do not miss their connections even if it means delaying some train) independent of the hump and assembly sequencing decisions.

We focus on the hump sequencing problem from operational and tactical level planning perspectives; we aim to layout principles of hump sequencing decisions. This is a first step in demonstrating that the effective yard capacities can be increased by using optimization-based tools even without proposing dramatic changes in the current methods railroads have been using for years.

3 Problem definition and computational complexity

We consider the following scheduling environment:

- A set of jobs $J = \{1, 2, ..., n\}$ is to be scheduled on a single machine.
- A job $j \in J$ has a processing time p_j and a release time r_j.
- A job $j \in J$ has a set of due dates which impose due date intervals (segments) with constant tardiness weights; let s_j be the number of due date segments. We define
  - d_{jl} as the due date corresponding to the beginning of the due date segment l, and
  - w_{jl} as the tardiness weight/cost of the due date segment l

  where $l \in \{1, 2, ..., s_j\}$.

We define the due dates individually for each job j. This is in order to generalize the modeling capability of this class of problems. A common due date scheme for all jobs, which may be the case for fixed transportation schedules, is a special case of this generic problem definition. Naturally, the first due date of a particular job is later than its release time (i.e., $r_j < d_{j1}$ for $j \in J$), and indices of the successive due dates imply a chronological order (i.e., $d_{j1} < d_{j2} < ... < d_{js_j}$ for $j \in J$). Since the tardiness weights are assumed to be stepwise increasing, we have $w_{j1} < w_{j2} < ... < w_{js_j}$ for $j \in J$.

Curry and Peters (2004), and Curry and Peters (2005) are the first studies to introduce the stepwise tardiness costs. They investigate a parallel machine scheduling problem with the additional machine assignment costs and set-up times. Considering the single-machine vs. multiple-machine environments, our problem in a sense seems to be a special case of the problem in Curry and Peters (2004). On the other hand, the scheduling problem in Curry and Peters (2004) considers the due date segments that are common to all jobs whereas we suppose that not all due date segments are applicable to all jobs which is also the case for the hump sequencing problem. In addition, we aim to construct a basis for further research on this new scheduling criterion by studying the single machine case with tardiness costs only.

The single-machine scheduling problems with minimum weighted total tardiness and minimum weighted number of tardy jobs discussed in Section 1 are similar to ours. Based on the $\alpha||\beta||\gamma$-classification scheme of Graham *et al.* (2000), these problems are denoted as $1\|r_j\|\sum w_jT_j$ and $1\|r_j\|\sum w_jU_j$, respectively. According to this classification scheme, the first entry corresponds to the number of machines, the second entry denotes the release times, and the third entry describes the scheduling criteria (the objective functions).

The major difference of our problem from the $1\|r_j\|\sum w_jT_j$ is that we have a stepwise increasing cost function instead of linearly increasing weighted tardiness cost function for each job. On the other hand, the major difference from the $1\|r_j\|\sum w_jU_j$ is that the number of due dates is more than one, among which the later ones have a larger penalty of missing than any of the earlier ones. According to the problem description in Curry and Peters (2004), we describe our problem as $1\|r_j\|\sum\sum w_{jl}U_{jl}$ where U_{jl} is equal to 1 for the earliest due date caught by the job completion time, and 0 for all the other due dates.

One of the most recent NP-hardness proofs for a single-machine scheduling problem is given in Du and Leung (1990) for $1\|\|\sum T_j$, which has remained as an open research question for a while after Lawler (1977) has developed a pseudo-polynomial time algorithm to solve it. The two problems associated with our problem of interest, $1\|\|\sum w_jT_j$ and $1\|\|\sum w_jU_j$, have been shown to be NP-hard much earlier; their being NP-hard trivially implies their release time versions to be NP-hard (see Brucker and Knust (2009) for a list of scheduling problems and their computational complexities). $1\|\|\sum w_jU_j$ is a special case of $1\|\|\sum\sum w_{jl}U_{jl}$, where the number of due dates, s_j, is only one for all jobs. Therefore, we can easily convince ourselves that $1\|\|\sum\sum w_{jl}U_{jl}$ is a generalization of $1\|\|\sum w_jU_j$. Thus, we have proved the following theorem.

Theorem 1 *$1\|\|\sum\sum w_{jl}U_{jl}$ and $1\|r_j\|\sum\sum w_{jl}U_{jl}$ are NP-Hard.*

4 Time-indexed formulations for single-machine scheduling problems

Time-indexed formulations have been extensively studied in machine scheduling research. Since our version of the scheduling objective function has not been studied extensively yet, we feel that it is critical to understand the performance of such formulations in obtaining optimal solutions, the quality of their relaxations and the effectiveness of heuristic algorithms based on these formulations. In this area of research, it was indeed more common to study enumerative and polyhedral approaches until the mid of 1990s. Even for the traditional scheduling criteria that were introduced almost 40 years ago now, the integer programming (IP) formulations and their relaxations have been studied extensively only in the last two decades. Nonetheless, heuristic (approximation) algorithms based on linear programming (LP) relaxations are now accepted as effective solution methods while lower bounds obtained from different relaxations are found to be strong (Savelsbergh *et al.* 2005). Some algorithms based on these relaxations were also proven to provide a constant factor approximations (e.g., see Hall *et al.* 1997).

The time-indexed formulations are based on a time discretization idea; time is divided into periods, where period t starts at time $t-1$ and ends at time t. The planning horizon of length T is composed of time periods $t \in \{1, 2, ..., T\}$. In this study, we focus on two main streams of time-indexed formulations: x_{jt}-formulation and y_{jt}-formulation. We note that the first problem formulation in Curry and Peters (2004) uses a slightly different formulation approach which assigns the jobs to due date periods with constant tardiness penalties. Our approaches depend on discretizing the planning horizon into time periods of constant length that are independent of due date segments of the objective function.

4.1 X-formulation: Completion time assignment

Savelsbergh *et al.* (2005) discuss the x_{jt}-formulation as a time-indexed formulation which generically models several single-machine scheduling problems; we refer to this type of formulation shortly as the X-formulation. Decision variables x_{jt} represent the completion time of jobs and are defined as follows:

$$x_{jt} = \begin{cases} 1, & \text{if job } j \text{ is completed at time } t; \\ 0, & \text{otherwise.} \end{cases}$$

In order to represent the stepwise tardiness scheduling criterion as an objective function, we define c_{jt} for $j \in J$ as follows:

$$c_{jt} = \begin{cases} 0, & 0 \leq t \leq d_{j1} - 1 \\ w_{j1}, & d_{j1} < t \leq d_{j2} - 1 \\ \vdots & \vdots \\ w_{js_j}, & d_{js_j} < t \leq T \end{cases}$$

The length of the planning horizon T is calculated as an upper bound by which all jobs are to be completed; it can be simply found as $T = \max_{j \in J}\{r_j\} + \sum_{j \in J} p_j$. Then, an exact IP formulation for the single-machine scheduling problem with stepwise increasing tardiness costs, denoted as X-IP, is as follows:

$$[X\text{-IP}] \text{ Minimize} \quad \sum_{j=1}^{n} \sum_{t=1}^{T} c_{jt} x_{jt} \tag{1}$$

$$\text{subject to} \quad \sum_{t=r_j+p_j}^{T} x_{jt} = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{j=1}^{n} \sum_{t'=t}^{t+p_j-1} x_{jt'} = 1 \qquad \forall t \in \{1, ..., T\} \tag{3}$$

$$x_{jt} \in \{0, 1\} \qquad \forall j \in J, \forall t \in \{1, ..., T\} \tag{4}$$

Objective function (1) minimizes the sum of tardiness costs overall jobs based on their completion time. Constraints (2) ensure that any job is completed earliest by its release time plus its processing time and latest by T. Constraints (3) are the capacity constraints ensuring that the machine can process at most one job at a time. (4) are the binary variable constraints. X-IP handles the job release times by restricting the set of possible completion times in constraint (2); it is easy to formulate the version with no release times by replacing (2) with $\sum_{t=p_j}^{T} x_{jt} = 1$. As noted by Savelsbergh *et al.* (2005), the objective function coefficients have to be modified to model the problems with different objective functions.

The LP relaxation of the X-formulation, X-LP, is obtained by replacing (4) with

$$x_{jt} \in [0,1] \qquad\qquad \forall j \in J, \forall t \in \{1, ..., T\}. \tag{5}$$

This type of relaxation is known to provide strong lower bounds when the objective function is the average weighted completion times of jobs, $1\|r_j\|\sum w_j C_j$, as well as the average weighted flow time, $1\|r_j\|\sum w_j F_j$ (see De Sousa and Wolsey 1992, van den Akker 1994, and Savelsbergh *et al.* 2005). However, this formulation, even as an LP relaxation, is not very easy to solve due to its size; van den Akker *et al.* (2000) has proposed Dantzig-Wolfe decomposition and column generation techniques to tackle the difficulties due to the size of the formulation.

4.2 Y-formulation: Preemption-based assignment

Another time-indexed formulation, known as y_{jt}-formulation (Savelsbergh *et al.* 2005), has been studied as a relaxation of $1\|r_j\|\sum w_j C_j$ (Hall *et al.* 1997, Phillips *et al.* 1998); this type of formulation will be denoted as the Y-formulation. The same idea of decision variables can be used to formulate first the preemptive version of $1\|r_j\|\sum\sum w_{jl} U_{jl}$, namely $1\|pmtn; r_j\|\sum\sum w_{jl} U_{jl}$. Traditionally, motivation in solving the preemptive version of the problem has been to use its solution as a lower bound for the non-preemptive version or to make use of the sequence of jobs in the preemptive schedule in order to construct a sequence for the non-preemptive schedule.

Formulating the preemptive version of $1\|r_j\|\sum w_j C_j$, $1\|pmtn; r_j\|\sum w_j C_j$, is straightforward as the completion time objective function can be easily mapped to a lower bounding objective function due to its linearity and monotonicity. However, in our case, the objective function is not linear. In addition, we cannot entertain any property of the objective function regarding the comparison of the rates of increase among a pair (or a set) of jobs. Accordingly, we slightly modify the idea of preemption in the following manner in order to formulate a preemptive scheduling problem:

- A job $j \in J$ is represented by p_j unit-time tasks; let $K_j = \{1, ..., p_j\}$ be the set of such unit-time tasks created from job j, and consider job j as the parent job of all unit-time tasks in K_j.
- A unit-time task $k \in K_j$ has the same release time as its parent job j.

- A unit-time task $k \in K_j$ mimics the tardiness cost structure of its parent job j in a partial manner; the ratio of its tardiness cost at any feasible completion time (which is also feasible for the parent job) is exactly $1/p_j$ of its parent's tardiness cost at the same completion time.

For the Y-formulation, we define

$$y_{jkt} = \begin{cases} 1, & \text{if task } k \in K_j \text{ of job } j \in J \text{ is processed at time } t; \\ 0, & \text{otherwise.} \end{cases}$$

Then, Y-PR, standing for the preemptive relaxation of the Y-formulation is

$$[Y\text{-PR}] \text{ Minimize} \quad \sum_{j=1}^{n} \sum_{k=1}^{p_j} \sum_{t=1}^{T} y_{jkt} \frac{c_{jt}}{p_j} \tag{6}$$

$$\text{subject to} \quad \sum_{t=r_j}^{T} y_{jkt} = 1 \qquad \forall k \in K_j, \forall j \in J \tag{7}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{p_j} y_{jkt} \leq 1 \qquad \forall t \in \{1, ..., T\} \tag{8}$$

$$y_{jkt} \in \{0, 1\} \qquad \forall k \in K_j, \forall j \in J, \forall t \in \{1, ..., T\} \tag{9}$$

Objective function (6) minimizes the sum of tardiness costs for all unit-time children of all jobs. Constraints (7) ensure that each unit-time job is processed only once after its release time; constraints (8) satisfy the machine capacity constraint, i.e., only one unit-time task is processed at a time. (9) are binary variable constraints. The set of unit-time tasks for a particular job j can be set up in an ordered fashion so that first unit-time task is released first and last unit time task ($p_j \in K_j$) is released last; we can modify (7) as

$$\sum_{t=r_j+(k-1)}^{T} y_{jkt} = 1 \qquad \forall k \in K_j, \forall j \in J$$

for a slightly more compact version of the original formulation.

We can show that the problem depicted with the Y-PR formulation is equivalent to an assignment problem with T matchings. To show this equivalence, we consider the assignment problem on a bipartite graph (N_1, N_2) where:

- each unit time task is represented by a node in N_1 implying that $|N_1| = \sum_{j \in J} p_j$;
- a time period $t \in \{1, ..., T\}$ is represented by a node in N_2 implying that $|N_2| = T$;
- c_{jt}/p_j is the assignment cost of a unit-time task in K_j to time period t.

Since $T \geq \sum_{j \in J} p_j$ by definition of a feasible schedule, we create additional $T - \sum_{j \in J} p_j$ dummy nodes in N_1 to have a feasible perfect matching of size T. The assignment problem has a polynomial complexity in its input size (i.e., number of matchings). However, the input of the scheduling problem does not include T. Therefore, T is pseudo-polynomial with respect to the input data of the scheduling problem depicted with the Y-PR formulation; the problem can be solved in pseudo-polynomial time. Hence, we can state the following result.

Theorem 2 *The scheduling problem depicted with the formulation Y-PR is pseudo-polynomially solvable.*

We have noted that the a unit time task only partially mimics the tardiness cost of its parent job as reflected by the tardiness weights in the objective function of the formulation Y-PR. Effectively, the tardiness weight coefficient of a unit time task in the objective function is always less than or equal to the actual tardiness weight of the parent job. Thus, the solution to the formulation Y-PR provides only a lower bound to the preemptive single-machine scheduling problem with stepwise increasing tardiness costs, namely $1\|pmtn; r_j\| \sum\sum w_{jl}U_{jl}$. Since $1\|pmtn; r_j\| \sum\sum w_{jl}U_{jl}$ is a relaxation of the non-preemptive problem, $1\|r_j\| \sum\sum w_{jl}U_{jl}$, the solution to the formulation Y-PR also provides a lower bound to $1\|r_j\| \sum\sum w_{jl}U_{jl}$. In order to obtain a non-preemptive version of the Y-formulation, Y-NP, that is equivalent to X-IP, we add the constraints:

$$y_{jkt} = y_{j,k+1,t+1} \qquad \forall k \in \{1, ..., p_j - 1\}, \forall j \in J, \forall t \in \{1, ..., T\} \tag{10}$$

which require that the unit-time children of a job are processed subsequently one after each other so that the job is processed in its entirety in a non-preemptive fashion. Then, the complete Y-NP formulation is:

$$[Y\text{-NP}] \text{ Minimize} \qquad \sum_{j=1}^{n}\sum_{t=1}^{T} y_{jp_jt}c_{jt} \tag{11}$$

$$\text{subject to} \qquad (7) - (10)$$

where the objective function (11) has non-zero coefficients for only those variables that represent the processing of the last unit-time task of a parent job j (unit-time task $p_j \in K_j$). As constraints (10) require that the unit-time jobs are processed in an ordered fashion according to their indices, the non-zero cost coefficient of the last unit-time task is equivalent to that of its parent job.

We also observe that the preemptive problem depicted with formulation Y-PR is equivalent to the non-preemptive version of problem with unit-time jobs (i.e. each parent job has only one child represented by a unit time task). Therefore, the following result is easily proven.

Theorem 3 $1\|p_j = 1; r_j\| \sum\sum w_{jl}U_{jl}$ *is pseudo-polynomially solvable.*

4.3 Background on the performance and quality of formulations

In this study, we compare the two formulations with respect to three performance measures: (i) the quality of lower bounds obtained from their relaxations, (ii) the computational effort to obtain the relaxation lower bounds and (iii) the computational effort to find exact optimal solutions. In this respect, we compare formulations X-LP and Y-PR for the lower bound performance; we compare formulations X-IP and Y-NP for the exact (optimal) solution performance. Another dimension of assessment that has been historically exploited in the scheduling literature is the closeness of the relaxation solutions to the optimal or near-optimal solutions. Rather than only assessing the percentage lower bound gaps, we are also concerned with the closeness of a feasible schedule imposed by relaxed solutions to the feasible schedule determined by the optimal (or near-optimal) solutions. In other words, we investigate how well the relaxed sequence of jobs implies the optimal sequence of jobs.

As mentioned earlier, the quality of lower bounds for the single-machine scheduling problems with completion time and flow time objective functions has been recently evaluated by Savelsbergh *et al.* (2005). In a similar fashion, we are more concerned with the quality rather than the computational effort with respect to different relaxation lower bounds. Indeed, we are aware that both types of relaxations may result in impractically large formulations which may require more computational effort than practically reasonable. The computational analysis regarding the quality of lower bounds in this study is a starting point to understand the effectiveness and applicability of modeling techniques that have been attempted for problems with previously known scheduling criteria. Indeed, the results of this study will also determine the direction of future research that may be more focused on improving the computational efficiency in obtaining these lower bounds.

It is naturally expected that both of the exact formulations, X-IP and Y-NP, will lead to computational challenges in obtaining optimal solutions for sufficiently large problems as the problem has been established to be NP-hard. Nonetheless, we intend to understand the difference and the level of difficulty between the two formulations at least to a reasonable extend that will identify the better one. In addition, we study a set of algorithms by which we would like to understand how close we can get to optimal solutions with much less computational effort than required by the exact formulations.

5 Heuristic algorithms

In this section, we develop two classes of heuristic algorithms for $1\|r_j\|\sum\sum w_{jl}U_{jl}$. The first class of algorithms relies on a relaxed solution of the algorithm. A relaxed solution of the problem is used to construct a feasible solution. Algorithms proposed in this study uses the LP relaxation of X-IP, X-LP, for the reasons which will be clear once the computational results for the relaxation lower bounds are studied. The second class, on the other hand, relies on a combinatorial approach which starts from an initial feasible solution and searches for improved solutions by enumerating the neighboring solutions of the initial solution.

5.1 LP-based sequence construction

The LP relaxation of the problem, X-LP provides fractional solutions to the time indexed formulation X-IP. The idea is to transform the partial assignment of jobs to a set of completion times with an integral assignment of each job to a single completion time. For $1\|r_j\|\sum w_jC_j$ and $1\|r_j\|\sum w_jT_j$, this class of algorithms has been first introduced by Phillips *et al.* (1998). Later, different variations have been proposed for problems with the same scheduling criteria. Other researchers (Hall *et al.* 1997, van den Akker *et al.* 2000) have proposed different variations of this type for $1\|r_j\|\sum w_jC_j$; some of these algorithms have been proven to provide constant-factor approximations. Savelsbergh *et al.* (2005) provide a comprehensive experimental study that compare the quality of such algorithms empirically as well. In this study, we develop two algorithms that use a relaxed solution information to construct a feasible sequence solution.

The first algorithm makes use of the notion of an α-point. The α-point of job $j \in J$, $0 \leq \alpha \leq 1$, is defined to be the first point in time, in the solution to a relaxation, at which an α fraction of job j has been completed. As the relaxed solution honors the release times, the α-point solution automatically honors these constraints. We refer to this class of algorithms as LPA-α which finds a feasible sequence based on α-points of jobs for a predetermined value of α. The steps of the algorithm, LPA-α, is as follows:

Step 1. Solve the formulation X-LP to obtain fractional assignment of jobs; let $\bar{x}_{jt}$ represent the values of the decision variables in the solution, $\forall j \in J, \forall t \in \{1, ...T\}$.

Step 2. For each job j, let $\bar{t}_j$ be the completion time corresponding to its α-point where

$$\bar{t}_j = \underset{t\in\{1,\ldots,T\}}{\operatorname{argmin}} \left\{ \sum_{t'=1}^{t} \bar{x}_{jt'} = \alpha \right\}.$$

Step 3. Order the jobs by their $\bar{t}_j$'s; schedule the jobs accordingly.

The choice of the value of α might be a critical factor in studying the performance of this class of algorithms. However, we cannot entertain any fact regarding the value of α to claim if larger (or smaller) α leads to better quality solutions. We take this consideration into account in our experimental study, and provide results for a set of different α values.

The second approach that constructs a sequence of jobs using the relaxed solution information has been first developed for total completion time problems. This approach calculates a completion time guess based on the partial completion time values in the relaxed solution. The steps of the algorithm are similar to that of LPA-α, but $\bar{t}_j$ of *Step 2* is calculated as $\bar{t}_j = \left\{ \sum_{t'=1}^{T} t\bar{x}_{jt'} \right\}$. This approach is referred to as 'schedule-by-completion-time' in the scheduling literature. It is shown by Hall *et al.* (1997) to be a constant factor approximation for $1||r_j||\sum w_jC_j$. van den Akker *et al.* (2000) empirically demonstrates that it is also very effective in practice. We refer to this approach as LPT in the rest of this study.

5.2 k-exchange neighborhood (k-opt) algorithms

This class of algorithms have been proposed for various difficult-to-solve combinatorial optimization problems. Yet, the single-machine scheduling literature has mostly relied on polyhedral approaches in the past, and more recently on the LP-based approaches. This is probably due to the fact that the traditional scheduling objective functions as we know are better behaving compared to a stepwise increasing function with some type of linearity or monotonicity of scheduling costs with respect to the completion time of jobs. On the other hand, the k-exchange neighborhood methods have been shown to be quite successful for problems which has less structured and worse behaving objective functions such as the infamous Traveling Salesman Problem. Our intentions in studying this class of algorithms for our problem is two-fold:

- to evaluate the quality of approaches that are independent of the mathematical formulation of the problem, but depend on combinatorial enumeration; and
- to provide alternative solution approaches whose worst case complexity are known and polynomial as opposed to the exponential complexity of LP relaxations or pseudo-polynomial complexity of the preemption-based relaxations.

The k-exchange neighborhood algorithm starts from an initial feasible solution of the algorithm. Each iteration visits the jobs according to the current sequence in an orderly fashion, which is called a complete pass. A job visit is comprised of searching the neighboring solutions where each neighborhood solution is obtained by exchanging the order of a list of subsequent jobs; the size of the list is determined by k. For instance, if $k = 3$, visiting the job in the i^{th} position, the algorithm compares six different possible solutions obtained by exchanging the order of jobs in positions i, $i + 1$ and $i + 2$ according to the current schedule. Comparing the set of neighboring solutions, the algorithm selects the solution with the minimum objective function value (or the maximum saving according to the current objective function value), and moves on to visit the current next job in the schedule.

The success of k-exchange algorithms depend on two factors: the initial solution and the size of neighborhood (i.e., the value of k). Different approaches can be used to select an initial solution. For instance, when release times are involved, the initial solution can be obtained by scheduling the jobs by their release times in a FIFO (first-in-first-out) order. Searching larger neighborhoods is hypothetically expected to improve the quality of solution, since the solutions enumerated by the larger neighborhood also contains those searched by a smaller neighborhood. Yet, it may not always be worth the effort to search larger neighborhoods. In our computational study, we test the performance of this class of algorithms with respect to different choices of the neighborhood size.

Algorithmic complexity. We first analyze the computational complexity of the 2-exchange algorithm for a single complete pass of the job list, which corresponds to an iteration of the algorithm, based on the following facts:

- a job visit that involves changing the schedule positions of two subsequent jobs and computing the new completion times accordingly has a complexity of $O(n)$;
- an iteration that is a complete pass over the scheduled job list is composed of $(n-1)$ job visits.

Therefore, a complete pass for 2-exchange can be performed in $O(n^2)$ time.

For larger values of k, we can analyze the complexity based on the analysis of 2-exchange. For instance, if $k = 3$, the number of neighboring solutions at a job visit is six instead of two of the 2-exchange. Then, the complexity of a 3-way exchange is larger then 2-exchange only by a constant factor. Since a complete pass has at most $n-2$ job visits, the final complexity is still $O(n^2)$.

It is important to note that k-exchange algorithm may be performed with multiple passes. One practical approach is to continue performing passes until no further improvement is possible.

6 Computational experiments

In this computational study, we discuss the following issues to investigate a fairly new problem:

- a comparative analysis of the IP formulations in terms of the computational effort for solving the problems to optimality;
- performance of the lower bounding techniques including
  - the strength of lower bounds in terms of the percentage gap from the optimal (best known) solution,
  - computational effort to solve the relaxed problems, and
  - closeness of the partial (relaxed) lower bounding solutions to the optimal (or near-optimal) solutions;
- quality of the upper bounds obtained by the heuristic algorithms and a comparative analysis of the computational effort among these algorithms.

Nonetheless, our main concern is to investigate the applicability of the proposed methods to solve our major motivation problem in the railroad context. In particular, we would like to answer the question ‘Are these techniques capable to solve the real-life hump sequencing problems?’ Furthermore, we would like to understand if these techniques can be extended to solve larger problems that contain such scheduling problems as a subproblem. In this respect, it is critical to understand not only the quality of solutions but also the computational effort in obtaining these solutions.

We organize this section in two main parts. The first part is concerned with an experimental study based on a set of randomly generated problems. This is the backbone of our computational analysis as this study is based on studying a single-machine scheduling problem with a fairly new scheduling criterion that has not been explored extensively. The second part is concerned with our practically relevant scheduling problem in the yard operations context where we present results with real-life data.

6.1 Experimental design

The machine scheduling literature has consistently generated problem sets that have been subsequently used by various researchers. In addition to a major data requirement such as the processing times of jobs, our problem definition shares some of the common features that have been considered in previous studies such as the release times and due dates. Regarding the processing times of jobs and release dates, we make use of the previous approaches in generating random data. In terms of the due dates, we need to extend the current approaches to generate multiple due dates for jobs; we should also consider generating tardiness costs for each due date of each job.

The foremost difference of stepwise tardiness problems from the traditional tardiness problems is a job's having multiple due dates. Therefore, we need to carefully account for this issue in generating the random problems. The number of due dates and how the due dates are shared by different jobs might play a significant role in generating random instances, not to mention how tardiness costs are generated for the set of due dates of a job.

Generating the random problem instances, we would like to capture the most generic form of the problem. Therefore, we generate problems whose jobs have their own due dates; due dates are individually generated for each job. Another issue in generating the due dates is related with the distribution of the due dates over the time line. The due dates may be distributed with identical length of time intervals among each other, or not. For instance, the due dates of a scheduling problem in the production planning context as discussed in Section 1 might be considered as equally spaced along the planning horizon as they are implied by the time points at the end of a day or at the end of a shift. On the other hand, the hump sequencing problem does not have this feature as the due dates are implied by the departure time of outbound trains which are distributed during the day based on the train schedules. To make the problem structure more generic, for a particular job given with the number of its due dates, we randomly distribute the due dates over the time line.

For a class of problems with a given distribution scheme of due dates along the time line, we generate instances according to four parameters:

- number of jobs (n),
- a distribution parameter for the random generation of job processing times (p),
- number of due dates (d), and
- a distribution parameter for the random generation of tardiness weights for each due date (w).

Although the stepwise tardiness scheduling problems are quite different than those we can find in the scheduling literature, we generate problem instances that resemble to those found in the literature for other scheduling problems. The number of jobs (n) is chosen from $\{20, 50\}$. The processing times are generated from a uniform distribution with p chosen from $\{10, 20, 40, 100\}$. Given the set of jobs specified with their processing times, the release times are generated according to an exponential arrival scheme. We allow the latest release time to occur at $\sum\limits_{j \in J} p_j$. Since $T = \sum\limits_{j \in J} p_j + \max\limits_{j \in J}\{r_j\}$, T is bounded by $2\sum_{j\in J} p_j$ from above. Regarding the number of due dates for jobs, we consider two types of problems; one type contains problems that has 10 due dates while the other one contains jobs with 20 due dates. For a particular job $j \in J$ given with its release time r_j, processing time p_j and number of due dates (10 or 20), its due dates are generated from $[r_j + p_j, T]$ with the same method we generate the job release times. The tardiness weight to be assigned to a due date segment is equal to the sum of the tardiness weight of its preceding due date and a random number generated from the uniform distribution $U(0, 10)$. For a particular job given with its number of due dates and the due dates, we start by generating a $U(0, 10)$ for its first due date segment. The tardiness weight for the second due date segment is obtained by adding the weight for the first segment with a new random $U(1, 10)$ number. We continue in this fashion until we generate the weight for the last due date segment of the job. As a result, we have non-decreasing due date weights that increment at the predetermined due dates of the job.

For the computational study, we generate 10 instances with 10 due dates and 10 instances with 20 due dates for a set specified with the number of jobs (n) and the distribution parameter p for processing times. A set of 20 problem instances generated this way is designated by a pair $n.p$ denoting the number of jobs and the processing time distribution parameter. Since we have two choices for n and four choices for p, we have eight different sets of problems with 20 instances generated for each set.

The computational tests are performed on a Pentium IV PC with 3.2 G Hz RAM. We used CPLEX 9.0 with Concert Technology 12 to solve the linear (integer) programming formulations. All computational times are reported in seconds.

6.2 Exact integer programming formulations

We first examine the performance of the two formulations in terms of the computational effort and practicality in obtaining optimal solutions. Our preliminary results identified that it is almost impossible to obtain optimal solutions and even an integer feasible solution for sufficiently large problem instances in a reasonable amount of time. Therefore, we have set some predetermined time limits based on the size of the problem instances for CPLEX to run based on our preliminary observations. Our analysis does not only identify which formulation takes shorter to find the optimal solution but also explores the formulations' success in finding optimal and feasible solutions in the predetermined time limits.

In Table 1, the 'Optimal Count' and 'Feasible Count' columns show (for each formulation) the number of problems (out of 20 problems in each row) that can be solved to optimality and feasibility, respectively. Note that the time limit has been set to 3600 and 7200 seconds respectively for the sets 50.40 and 50.100, and 1800 seconds for the remaining problems. X-IP clearly outweighs the success rate of Y-NP in reaching feasible and optimal solutions particularly for the larger problem instances. Recalling that the size of Y-NP depends not only on the number of jobs and the length of the planning horizon but also on the job processing time data, this is reasonable. With respect

Table 1: Results for comparing the performance of the exact integer programming formulations.

	Optimal count		Feasible count		Optimal Time	
$n.p$	X-IP	Y-NP	X-IP	Y-NP	X-IP	Y-NP
20.10	10	10	10	10	0.2	0.5
20.20	10	10	10	10	1.8	2.1
20.40	10	10	10	10	2.4	7.9
20.100	10	1	10	1	152.5	44.9
50.10	10	10	10	10	35.3	11.9
50.20	9	9	10	10	13.0	17.6
50.40	10	0	10	0	51.1	n/a
50.100	2	0	6	0	3047.9	n/a

to particular problem instances, X-IP finds the optimal (feasible) solution to all instances for which Y-NP can reach an optimal (or feasible) solution. Reported solution times include information only for the instances which are solved to optimality by either of the formulations. Therefore, summarized results may not be directly helpful since the number of optimal instances are not identical in each cell. Yet, we observe that the solution times with X-IP are much less than those with Y-NP with respect to individual instances although this may not be clear for each problem set. For instance, for the set 20.100, the average solution time with Y-NP is less than that of X-IP since Y-NP solution time for this set contains a single instance out of 20 problems in the set.

6.3 Lower bounds and relaxed schedules

We compare the performance of the relaxed formulations, X-LP and Y-PR, with respect to the strength of the lower bounds provided by the relaxations and the computational effort in obtaining them. The lower bound gaps are computed according to the objective function of the best known feasible solution, IP_{best}, as $(\text{IP}_{best} - \text{LB})/\text{LB}$ where LB denotes the value of the lower bound, and are reported as percentages. No percentage gap can be provided for instances with no feasible solution from the exact programming formulations.

Results in Table 2 show that the lower bounds obtained from the relaxation X-LP are stronger than those obtained from the preemptive relaxation Y-PR. Note that for the set 50.100, even the relaxed solutions cannot be obtained with Y-PR; the Y-PR formulation becomes extremely large because of the magnitude of job processing times and the instances in this set cannot be solved due to memory related problems. Regarding the relaxation solution times reported in Table 2, it is clear that the computational effort is much less with X-LP relaxation.

As discussed earlier in Section 4, another issue regarding the relaxed solutions is concerned with the closeness of the relaxed solution schedules to the optimal schedules (or the best known schedule when optimal is not available). To determine the schedules imposed by the relaxed solutions, we use two alternative methods for both types of relaxed formulations. The first method looks at the first partial completion time inferred from the relaxed solution (X-LP-First, Y-PR-First) while the

Table 2: Performance of the relaxed formulations.

	Lower Bound Gap (%)		Solution Time	
n.p	X-LP	Y-PR	X-LP	Y-PR
20.10	0.53	4.39	0.1	0.4
20.20	0.41	4.16	0.1	2.1
20.40	0.66	5.61	0.4	11.6
20.100	0.79	5.66	2.3	72.4
50.10	0.21	1.70	0.4	3.2
50.20	0.21	1.84	1.4	6.4
50.40	0.13	1.85	2.7	40.0
50.100	0.69	n/a	79.7	n/a

Table 3: Percentage closeness ratios of the relaxed schedules to optimal (or best) schedules.

	X-LP-First	X-LP-Last	Y-PR-First	Y-PR-Last
n.p	(%)	(%)	(%)	(%)
20.10	52.5	57.5	36.0	44.0
20.20	50.5	47.5	37.0	44.0
20.40	51.0	49.5	32.5	36.0
20.100	52.5	49.5	35.0	60.0
50.10	26.4	31.2	21.6	28.4
50.20	27.2	24.4	21.2	24.2
50.40	32.2	27.0	n/a	n/a
50.100	14.2	10.2	n/a	n/a

second method looks at the last partial completion time inferred from the relaxed solution (X-LP-Last, Y-PR-Last). For a relaxed solution schedule, we measure the closeness as follows: if a job is in the same position in both relaxed and optimal schedules, it is counted as a closeness score of 1. Then, for a relaxed solution we sum up the closeness score over all jobs. Closeness of the relaxed schedule is calculated as the percentage ratio of sum of the closeness scores to the number of jobs. The results are reported as percentage ratios in Table 3.

Results in Table 3 is not as conclusive as the previous results to point to one of the relaxed solutions as better than the other one. A noteworthy outcome is that for both formulations the closeness factors are significantly smaller for problems with 50 jobs compared to those with 20 jobs. In addition, the results for the set 50.100 show how unrelated the schedules can be when the degree of partialness in the solution increase with the job processing times as well as number of jobs. It is also interesting to note that the strength of the lower bounds does not necessarily worsen for these problems in comparison to the smaller instances (see Table 2).

6.4 Heuristic algorithms

We have presented two classes of heuristic algorithms for $1||r_j||\sum\sum w_{jl}U_{jl}$ in Section 5. Computational experiments on the performance of exact solution methods clearly show that it is not always practically possible to obtain optimal solutions (or even feasible solutions). We now investigate the proposed heuristic algorithms for the quality of solutions obtained and the computational effort in obtaining these solutions.

The first approach in the class of LP-based heuristic algorithms, LPA-α, relies on the notion of α-point of a job as discussed earlier. Different variations of this class is tested based on different choices of α. We test this class of algorithms for the values of α from $\{e, 0.25, 0.5, 0.75, 1\}$ where $e > 0$ and sufficiently small. The second approach, LPT, based on the completion time estimate does not have any variations. The k-exchange neighborhood search algorithms are tested for three values of k: 2, 3 and 4.

Our experiments show that for both classes of algorithms, the computational effort to reach a near-optimal solution is almost negligible assuming that the relaxed solutions from the X-LP formulation are already known for the LP-based algorithms. On the other hand, we cannot determine a best/good choice of the algorithm parameters, α for the LP-based algorithm class and k for the class of neighborhood algorithms. Yet, 2-exchange algorithms have found worse solutions than both 3-exchange and 4-exchange algorithms for all the instances in our test problems. Therefore, we implemented the following ideas:

- LPA-α-best finds the α-point solutions for five different values of $\alpha \in \{e, 0.25, 0.5, 0.75, 1\}$; it selects the one with the minimum objective function value as the final solution.
- 3/4-exchange algorithm first performs 3-exchange passes until no further improvement is possible. Then, starting from the final solution of the 3-exchange passes, it performs 4-exchange passes until again no further improvement is possible to obtain the final solution.

Our experiments show that the final solution can be obtained in less than 0.1 seconds for both classes of algorithms. Therefore, in Table 4, we only present results regarding the quality of the heuristic algorithms excluding the computational times. We note that the LP-based algorithms uses the relaxed solution from the X-LP formulation. Therefore, we should consider the time to obtain the relaxed solution (from Table 2) in addition to 0.1 seconds of empirically worst case run-time for the LPA-α-best algorithm.

In Table 4, we report the percentage gap of the algorithm solutions from the optimal (or best known) solution obtained by the exact solution methods. We also report the results for the best solution obtained by any of the LP-based algorithms (including the results for five different values of α for LPA-α and LPT) in the column LP-best; the last column 'Difference' shows the average percentage difference between the quality of the 3/4-exchange and LP-best for all instances in a problem set. The last row shows the average values for each column. Negative percentage values indicate that the result obtained by the heuristic algorithm is better than the best solution obtained by the exact formulations with CPLEX in limited run times.

Table 4: Results for the heuristic algorithms.

	Optimality Gap (%)				Difference
$n.p$	LPA-α-best	LPT	LP-best	3/4-exchange	(%)
20.10	1.67	3.82	1.67	3.59	1.92
20.20	1.64	4.35	1.29	2.27	0.99
20.40	2.45	5.96	1.92	1.87	-0.05
20.100	1.64	5.15	1.57	3.71	2.14
50.10	2.05	3.69	1.85	5.60	3.75
50.20	1.52	4.15	1.51	5.78	4.27
50.40	0.65	3.94	0.65	3.11	2.46
50.100	-4.05	-2.00	-3.85	-0.58	3.27
Average	0.94	3.63	0.83	3.17	2.34

The results show that the LP-based algorithm class reaches near-optimal solutions in the vicinity of 2% in the worst case and 1% on average. For the larger problem instances, it is almost impossible to obtain a good quality solution with the exact solution methods. LPA-α type algorithm produces better results when compared to LPT algorithm on average. However, this is not true for all instances on individual basis. For some instances, LPT may reach a better solution than that of LPA-α as column LP-best shows that averages combining the results from all LP-based algorithms are better than that of a single type. Considering that the 3/4-exchange algorithm has a much less computational effort than the LP-based algorithms, the results are also promising for neighborhood search type algorithms. In addition, we cannot necessarily conclude that the LP-based class always finds a better solution the k-exchange class (e.g. see problem set 20.40).

6.5 Hump sequencing problems with real-life data

Our major motivation in this study is solving the hump sequencing problem which has a practical relevance in timeliness of railroad yard operations and in increasing the yard capacities as discussed in Section 2. We have obtained a one week data set from a major US railroad company for our experimental study. The data belongs to a medium-size classification yard. From a practical planning perspective, yard masters need a tool to schedule their hump processes on a daily basis. This tool can be mainly used at the start of each shift during the day to schedule the humping of inbound trains including those which have already arrived and those which are being expected in the next 24 hours. This tool is required also when major schedule disruptions and unexpected delays on both the inbound side and outbound side of the yard occur, which will eventually modify the arrival times and connection times of the trains. Therefore, the efficiency of the tool is as crucial as the accuracy and quality of the solutions.

In this section, we study seven one-day problems from our weekly data set; we select the most congested day (with 19 inbound trains and 18 outbound trains) to investigate the results in detail. In addition to the quality and efficiency of the solutions obtained by the X-IP formulation, we also

Table 5: Results for a one-day problem with real-life data with 19 inbound trains and 18 outbound trains.

Time Unit (minutes)	Inspection Capacity	FIFO	Optimal	Improvement	Solution Time
1	1	124	53	2.34	15.15
1	2	102	16	6.38	24.11
5	1	124	53	2.34	0.87
5	2	102	16	6.38	0.97
10	1	124	53	2.34	0.37
10	2	102	16	6.38	0.41

investigate the sensitivity of the methods for the choice of time discretization which significantly changes the size of the exact formulations and relaxed formulations. The original arrival, processing, and departure times are given in minutes; we construct three different versions of each one-day problem by assuming a time discretization of one, five, and 10 minutes. We note that five-minute discretization is practically found to provide more than sufficient accuracy for the most of the railroad operations. Nonetheless, we aim to understand how the accuracy of solutions compensate for the additional computational effort.

Creating the problem instances from the real-life data, we also need to account for another process in the context of yard operations: the inbound inspection of trains. For safety regulations, an inbound train needs to be inspected by the yard inspectors as soon as possible upon its arrival. This operation, however, should be performed on a FIFO basis; inbound trains are inspected in the same order as they arrive. Therefore, there is not much of a scheduling problem regarding the inspection process. However, the inspection process effects the earliest time a train can be humped which is the release time of a job with respect to our problem definition. The inspection time varies among the trains based on the number of cars on the train and the physical length of the train. The number of trains that can be inspected simultaneously depends on the availability of inspection staff during a shift/day, and the simultaneous inspection capacity is usually either one train or two trains for a typical US railyard. The inspection capacity of the yard affects the rate of congestion for the trains waiting to be humped, which changes the rate of job releases with respect to our problem definition. For each one-day problem data, we create two versions of the release times: assuming an inspection capacity of either one or two. Coupled with the choice of time discretization, we create a total of six instances for a one-day hump sequencing problem. In Table 5 and Table 6, the first and second columns denote the choice of time unit for discretization and the yard inspection capacity, respectively. Table 5 shows the results for the most congested day from our weekly data set; Table 6 shows averages over seven one-day problems.

The objective function of the hump sequencing problem minimizes the number of cars that miss their outbound connections. We compare the solutions obtained by the IP formulation with a very-commonly used hump sequencing method by the yard masters, namely the FIFO rule. Table

Table 6: Results for one-day problems with real-life data of a given week (averages over 7 instances).

Time Unit	Inspection	Improvement Ratio			Solution Time	
(minutes)	Capacity	Min	Average	Max	Average	Max
1	1	2.34	3.74	5.89	12.46	15.62
1	2	4.33	6.87	10.50	14.81	24.11
5	1	2.34	3.74	5.89	0.77	0.90
5	2	4.33	6.63	10.50	0.78	0.97
10	1	2.34	3.74	5.89	0.35	0.60
10	2	4.33	6.58	10.00	0.34	0.51

5 compares the number of cars that miss their outbound connections for the two solutions in each row: solution obtained from the natural humping order of trains (in FIFO order) and the solution obtained from the exact solution of the problem (with the X-IP formulation). The column 'Improvement Ratio' shows the ratio of the number of missing cars in FIFO hump order to that of the optimal solution; the larger the ratio is the more improvement optimal solution provides over the natural hump order. The results clearly show that there is a significant difference between the two solutions. In general, solution times are even better than what the yard master would hope for. As expected, the solution times increase as the time unit for discretization gets smaller since the IP formulation becomes larger. Nonetheless, the choice of time unit for discretization does not affect the accuracy of the solution. However, we should note that the accuracy comparison between different choices of time discretization is not conclusive only with this data set. Another noteworthy result is that the assumed inspection capacity is an important parameter that may affect the accuracy of the solutions.

Table 6 shows results for the set of seven one-day problems. Results regarding the improvement ratio corresponds to the ratio between the FIFO humping and hump sequence obtained from the IP formulation as in Table 5. It shows that the number of cars that miss their outbound connections can be as much as 10 times of the optimal solution when the traditional FIFO order is used. It is also notable that the improvement potential is even greater when the yard inspection capacity is larger since the FIFO solution turns out to be even poorer as some trains can be ready for humping before the others that arrive earlier. Regarding the solution times, we observe that the choice of discretization as one minute can result in times that may not be always wanted. However, the accuracy of the solutions with larger time unit choices is not deteriorated; any problem in this data set can be solved to optimality in less than one second.

We also note that the results in Table 5 and Table 6, particularly regarding the solution times, are encouraging not only for the practical uses of this method in real-life hump sequencing. It also shows that the X-IP formulation can be used as an exact optimization subroutine in the context of a heuristic algorithm for the larger yard optimization problem which may require calling a hump sequencing solution procedure several times.

7 Concluding remarks

Studying the hump sequencing problem in the context of yard operations, we have identified a class of machine scheduling problems with a fairly new scheduling criteria that has a justifiable practical relevance in the context of traditional production planning and supply chain optimization. For this new scheduling problem, we have developed IP formulations based on traditional ideas from the scheduling literature. As the problem is shown to be NP-complete, we have also developed heuristic algorithms. The first class of heuristic algorithms is based on the relaxation of the exact IP formulations of the problem; we use some ideas that have been tried for other machine scheduling problems. The second class of heuristic algorithms, k-exchange or k-opt, has not been traditionally employed for machine scheduling problems. Both algorithms produce good-quality solutions based on our experimental data set, relaxation-based algorithms producing better results for the price of more computational effort. The k-exchange algorithms, on the other hand can produce good quality solutions in less than tenth of a second even for the largest instances.

Results for the hump sequencing problems with real-life data are even more encouraging. We observe that a one-day problem for a medium size US railyard can be solved to optimality in under one second. Our real-life data set shows that the time discretization does not deteriorate the accuracy or quality of the solutions. Most importantly, comparing the optimal solutions with the yard master's traditional FIFO rule, we observe that the room for improvement is quite large. The US railroads have not used methods based on operations research (OR) techniques to manage their yard operations yet. Our results show that desired yard capacity upgrades can be obtained by improving the operational efficiency first rather than expanding the physical infrastructure.

We believe that this study will encourage both practitioners and researchers to focus on developing OR-based decision support tools for yard operations planning. In addition, this will be a first step in exploring the stepwise tardiness problems with more complicated settings.

References

J. Curry and B. Peters, "Solving parallel machine scheduling problems with stepwise increasing tardiness cost objectives," Working Paper, Department of Industrial Engineering, Texas A&M University, 2004.

J.R. Birge and M.J. Maddox, "Bounds on expected project tardiness," Operations Research, **43**(5), pp. 838–850, 1995.

C.D. Martland, "PMAKE analysis: Predicting rail yard time distributions using probabilistic train connection standards," Transportation Science, **16**(4), pp. 476–506, 1982.

E.R. Petersen, "Railyard modeling: Part I. Prediction of put-through time," Transportation Science, **11**(1), pp. 37–49, 1977a.

E.R. Petersen, "Railyard modeling: Part II. The effect of yard facilities on congestion," Transportation Science, **11**(1), pp. 50–59, 1977b.

M.A. Turnquist and M.S. Daskin, “Queuing models of classification and delay in railyards,” Transportation Science, **16**(2), pp. 207–230, 1982.

C.F. Daganzo, R.G. Dowling and R.W. Hall, “Railroad classification yard throughput: the case of multistage triangular sorting,” Transportation Research Part A, **17**(2), pp. 95–106, 1983.

C.F. Daganzo, “Static blocking at railyards: Sorting implications and tracks requirements,” Transportation Science, **20**(3), pp. 186–199, 1986.

C.F. Daganzo, “Dynamic blocking for railyards: Part I. Homogeneous traffic,” Transportation Research Part B, **21**(1), pp. 1–27, 1987a.

C.F. Daganzo, “Dynamic blocking for railyards: Part II. Heterogeneous traffic,” Transportation Research Part B, **21**(1), pp. 29–40, 1987b.

E.R. Kraft, “A Hump sequencing algorithm for real time management of train connection reliability,” Journal of the Transportation Research Forum, **30**(4), pp. 95–115, 2000.

E.R. Kraft, “Priority-based classification for improving connection reliability in railroad yards–part I: Integration with car scheduling,” Journal of the Transportation Research Forum, **56**(1), pp. 93–105, 2002a.

E.R. Kraft, “Priority-based classification for improving connection reliability in railroad yards–part II: Dynamic block to track assignment,” Journal of the Transportation Research Forum, **56**(1), pp. 107–119, 2002b.

J. Curry and B. Peters, “Rescheduling parallel machines with stepwise increasing tardiness,” International Journal of Production Research, **43**(15), pp. 3231-3246, 2005.

R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, “Optimization and approximation in deterministic sequencing and scheduling: A survey,” Annals of Discrete Mathematics, **5**, pp. 287–326, 1979.

J. Du and J.Y.-T. Leung, “Minimizing total tardiness on one machine is NP-hard,” Mathematics of Operations Research, **15**(3), pp. 483–495, 1990.

E.L. Lawler, “A ‘Pseudopolynomial’ algorithm for sequencing jobs to minimize total tardiness,” Annals of Discrete Mathematics, **1**, pp. 331–342, 1977.

P. Brucker and S. Knust, “Complexity results for scheduling problems,” Available online at: http://www.informatik.uni-osnabrueck.de/knust/class/ (accessed 1 March 2011).

M.W.P. Savelsbergh, R.N. Uma and J. Wein, “An experimental study of LP-based approximation algorithms for scheduling problems,” INFORMS Journal on Computing, **17**(1), pp. 123–136, 2005.

L.A. Hall, A.S. Schulz, D.B. Shmoys and J. Wein, “Scheduling to minimize average completion time: off-line and on-line approximation algorithms,” Mathematics of Operations Research, **22**(3), pp. 513-544, 1997.

J.P. De Sousa and L.A. Wolsey, “A time-indexed formulation of non-preemptive single-machine scheduling problems,” Mathematical Programming, **54**(3), pp. 353-367, 1992.

M. van den Akker, “LP-based solution methods for single-machine scheduling problems,” Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, The Netherlands, PhD Thesis, 1994.

M. van den Akker, C.A.J. Hurkens and M.W.P. Savelsbergh, “A time-indexed formulation for single-machine scheduling problems: Column generation,” INFORMS Journal on Computing, **12**(2), pp. 111-124, 2000.

C. Phillips, C. Stein and J. Wein, “Minimizing average completion time in the presence of release dates,” Mathematical Programming B, **82**(1-2), pp. 199-224, 1998.